Programski jezici

http://www.programskijezici.matf.bg.ac.rs/

Univerzitet u Beogradu
Matematički fakultet

Programske paradigme

Materijali za vežbe

Nastavnik: Milena Vujošević Janičić
Asistenti: Nemanja Mićović, Milica Selaković

Beograd
2018.

Priprema materijala:
dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu
Marjana Šolajić, asistent na Matematičkom fakultetu u Beogradu
Branislava Živković
Nemanja Mićović, asistent na Matematičkom fakultetu u Beogradu
Milica Selaković, asistent na Matematičkom fakultetu u Beogradu

Sadržaj

1

Skript programiranje

Potrebno je imati instaliran Python 2.7 na računaru.
Literatura:

(a) https://www.python.org/

(b) http://www.tutorialspoint.com/python

(c) https://wiki.python.org/moin/

Merenje performansi programskih jezika:

- https://benchmarksgame.alioth.debian.org/

1.1 Uvod, kolekcije, matematičke funkcije

1.1.1 Uvodni primeri

Zadatak 1.1 Napisati program koji na standardni izlaz ispisuje poruku *Hello world! :)*.

```
# Ovako se pisu komentari
#
# Pokretanje programa iz terminala:
#  $python  hello.py
#
print "Hello world! :)"
```

Zadatak 1.2 Napisati program koji za uneta dva cela broja i nisku ispisuje najpre unete vrednosti, a zatim i zbir brojeva, njihovu razliku, proizvod i količnik.

```
# Promenljive se dinamicki tipiziraju
# a = 45
# b = 67.45
# istina = True
# Niske su konstantne tj. nisu promenljive.
# To znaci da se menjanjem nekog karaktera u niski
# pravi nova niska u memoriji.
#niska = "I believe i can fly!"

# Ispis na standardni izlaz
# print a
# print b
# print a, b, istina

# Ucitavanje niske sa standardnog ulaza
print "\n------Ucitavanje sa standardnog ulaza------\n"
string_broj = raw_input("Unesite ceo broj: ")
a = int(string_broj) # vrsi se konverzija stringa u ceo broj, slicno: float, str

string_broj = raw_input("Unesite ceo broj: ")
b = int(string_broj)
```

```
22
niska = raw_input("Unesite nisku: ")
24
# Formatiran ispis
26 print "\n------Formatiran ispis------\n"
print "Brojevi: {0:d} {1:d}\nNiska: {2:s}\n".format(a,b,niska)
28
# Osnovne aritmeticke operacije:
30 # +, -, *, /, %, ** (stepenovanje)
zbir = a + b;
32 razlika = a - b;
proizvod = a * b;
34 kolicnik = float(a) / b;

36 print "Zbir {0:d}\nRazlika {1:d}\nProizvod {2:d}\nKolicnik {3:f}\n".format(zbir,
    razlika, proizvod, kolicnik)
```

Zadatak 1.3 Ako je prvi dan u mesecu ponedeljak napisati funkciju `radni_dan(dan)` koja kao argument dobija dan u mesecu i vraća tačno ako je dan radni dan. Napisati program koji testira ovu funkciju, korisnik sa standardnog ulaza u petlji unosi deset dana i dobija o poruku o tome da li su uneti dani radni ili ne.

```
# Funkcije
2 #
# def ime_funkcije(argumenti):
4 #     telo funkcije

6 def radni_dan(broj):
    # Osnovne logicke operacije:
8       # not, and, or

10      if broj % 7 == 1 or broj % 7 == 2 or broj % 7 == 3:
            return True
12      elif broj % 7 == 4 or broj % 7 == 5:
            return True
14      # Naredbi <<elif>> moze biti vise
        else:
16          return False

18 # Blokovi se ne ogranicavaju viticastim zagradama kao sto je u C-u
# vec moraju biti uvuceni tabulatorom.
20
i = 0
22 # Petlja
while i <= 10:
24      dan = raw_input("Unesite dan")
        dan = int(dan)
26      if radni_dan(dan):
            print "Uneti dan je radni dan"
28      else:
            print "Uneti dan je neradan"
30          i=i+1 # i++ ne postoji, moze ili ovako ili i+=1

32 # Naredba <<break>> iskace iz bloka, isto kao i u C-u
# Naredba <<pass>> je ista kao naredba <<continue>> u C-u
```

Zadatak 1.4 Napisati program koji na standardni izlaz ispisuje vrednost $6!$, $\log_5 125$ i pseudo slučajan broj iz opsega $[0, 1)$

```
1 # Matematicke funkcije

3 # Ukljucujemo modul <<math>>
import math
5
# U ovom moduli se nalaze brojne funkcije kao sto su:
7 #
# math.sqrt(broj)
9 # math.log(broj, osnova)
# math.sin(ugao_u_radijanima), math.cos(), ...
11 # math.exp(stepen)
# math.factorial(broj)
```

```
# i druge...
print "\n------Matematicke funkcije------\n"
print math.factorial(6)
print math.log(125, 5)

# Pseudo slucajni brojevi

# Ukljucujemo modul <<random>>
import random

# Funkcija random() vraca pseudo slucajan broj tipa float iz opsega [0.0, 1.0)
print "\n------Pseudo slucajni brojevi------\n"
print "Pseudo slucajan broj iz opsega [0.0,1.0)\n"
print random.random()

# Korisne funkcije:
#
# randint(a,b) - vraca pseudo slucajan ceo broj n iz opsega [a,b]
# choice(lista) - vraca pseudo slucajan element iz liste
#
```

Zadatak 1.5 Napisati program koji imitira rad bafera. Maksimalni broj elemenata u baferu je 5. Korisnik sa standardnog ulaza unosi podatke do unosa reči *quit*. Program ih smešta u bafer, posto se bafer napuni unosi se ispisuju na standarni izlaz i bafer se prazni.

```
# LISTA
#
# Notacija: [element1, element2, ...]
#
# Liste mogu sadrzati razlicite tipove podataka
# lista = [1,2,3.4,"Another brick in the wall",True,[5,False,4.4,'Layla']]

# Prazna lista
buffer = []
i = 0
while True:
    unos = raw_input()
    if unos == 'quit':
        break
    # Ubacivanje elementa na kraj
    buffer.append(unos)
    i += 1
    if i == 5:
        # Prolazak kroz listu
        for unos in buffer:
            print unos
            buffer = []
            i = 0
```

Zadatak 1.6 Korisnik sa standardnog ulaza unosi ceo broj n, a potom ciklično pomeren rastuće sortiran niz (pr. 56781234) koji ima n elemenata. Napisati program koji na standarni izlaz ispisuje sortiran niz bez ponavljanja elementa.

```
# Korisne funkcije za liste:
#
# list.remove(x) - izbacuje prvo pojavljivanje elementa x iz liste
# list.count(x) - vraca broj koliko puta se element x nalazi u listi
# list.index(x) - vraca indeks prvog pojavljivanja elementa x u listi
# len(lista) - vraca broj elemenata liste
# del lista[a:b] - brise elemente liste od pozicije a do b

n = int(raw_input("Unesite broj elemenata"))

elementi = []
# Funkcija <<range>>
#
# range(kraj)
# range(pocetak, kraj[, korak])

for i in range(n):
    element = raw_input()
```

```
    # Provera da li se element nalazi u listi
    if not element in elementi:
        # Ubacivanje elementa na odredjenu poziciju u listi
        elementi.insert(i, element)

k = 0
for i in range(1, n):
    # Pristupanje elementima liste
    if elementi[i - 1] > elementi[i]:
        k = i
        break

prvi_deo = elementi[0:k]
drugi_deo = elementi[k:]

# Nadovezvivanje dve liste
sortirani = drugi_deo + prvi_deo
print "Sortirani elementi"
for element in sortirani:
    print element
```

Zadatak 1.7 Napisati funkciju `max_list(lista)` koja vraća najveći element u listi listi. Napisati program koji testira ovu funkciju.

```
def max_list(lista):
    # Mozemo indeksirati liste unazad, pozicija -1 odgovara poslednjem elementu
    maximum = lista[-1]
    for element in lista:
        if maximum < element:
            maximum = element
            return maximum

lista = [[1,2,3], [1,2,5], ['abc','abc','abc'], ['abc', 'ab', 'abcd'], ['a','b','c']
    > ['a', 'b']]
print max_list(lista)
```

Zadatak 1.8 Napisati program za rad sa stekom.

- Definisati stek koji sadrži elemente $9, 8, 7$
- Dodati na stek elemente 6 i 5
- Skinuti sa steka element i ispisati ga na standardni izlaz

```
# Koriscenje liste kao stek strukture podataka
stek = [9,8,7]
# Operacija push je implementirana funkcijom append
stek.append(6)
stek.append(5)
print "\n------Ispisujemo stek------\n"
print stek
# Operacija pop je implementirana funkcijom pop
print "\n------Ispisujemo element dobijem funkcijom pop------\n"
print stek.pop()
print "\n------Ispisujemo znanje nakon pozivanja funkcije pop------\n"
print stek
```

Zadatak 1.9 Napisati program koji za uneti prirodan broj n ispisuje vrednosti funkcije x^2 u celobrojnim tačkama u intervalu $[0, n]$. Zadatak rešiti korišćenjem mape.

```
# KATALOG
#
# Katalog je kolekcija uredjenih parova oblika (kljuc, vrednost)
#
# Notacija: {kljuc:vrednost, kljuc:vrednost, ...}

mapa = {} # prazna mapa

n = int(raw_input("Unesite n: "))
```

```
10
for x in range(n):
12      mapa[x] = x**2

14  # Prolazak kroz mapu
    print "\n------Prolazak kroz katalog------\n"
16  for kljuc in mapa:
        print "f({0:s}) =  {1:s}\n".format(str(kljuc),str(mapa[kljuc]))
18
    # Korisne funkcije
20  #
    # map.keys() - vraca listu kljuceva iz kataloga
22  # map.values() - vraca listu vrednosti iz kataloga
    # map.has_key(kljuc) - vraca True/False u zavisnosti od toga da li se element
24  # sa kljucem kljuc nalazi u katalogu
```

Zadatak 1.10 Sa standardnog ulaza se unose reči do reči *quit*. Napisati program koji ispisuje unete reči eliminišući duplikate.

```
lista = []
2
while True:
4       unos = raw_input()
        if unos == "quit":
6           break
        lista.append(unos)
8
# Pravljenje skupa od liste
10  skup = set(lista)

12  for i in skup:
        print i
```

Zadatak 1.11 Napisati funkciju `min_torka(lista)` koja vraća najmanji element u torci torki. Napisati program koji ovu funkciju testira.

```
1   # Uredjene N-TORKE
    print "\n------Torke------\n"
3   # torka = ("Daffy","Duck",11)

5   def min_torka(torke):
        # Pristupanje elementima u torki
7       minimum = torke[0]
        for torka in torke:
9           # Ukoliko torke ne sadrze elemente istog tipa na istim pozicijama, i dalje ih
         mozemo porediti,
            # ali poredjenje se vrsi na osnovu imena tipa elementa leksikografski
11          # npr.  element tipa List < element tipa String < element tipa Tuple i slicno
            if minimum > torka:
13              minimum = torka
                return minimum
15
    torke = ((1,2,'a'),(1,2,'b'),([1,2,3], 'Bugs', 4), ([1,1,1], 'Bunny', 6),(1,2,['a','b
        ']),(1,2,'ab'))
17
    print min_torka(torke)
```

1.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.12 **Pogodi broj** Napisati program koji implementira igricu "Pogodi broj". Na početku igre računar zamišlja jedan slučajan broj u intervalu [0,100]. Nakon toga igrač unosi svoje ime i započinje igru. Igrač unosi jedan po jedan broj sve dok ne pogodi koji broj je računar zamislio. Svaki put kada igrač unese broj, u zavisnosti od toga da li je broj koji je unet veći ili manji od zamišljenog broja ispisuje se odgovarajuća poruka. Igra se završava u trenutku kada igrač pogodio zamišljen broj.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
POKRETANJE: pogodi_broj
 -------- IGRA: Pogodi broj ---------
 Unesite Vase ime:
 Milica
 Zdravo Milica. :)
 Zamislio sam neki broj od 1 do 100. Da li mozes da pogodis koji je to broj?
 Unesi broj
 50
 Broj koji sam zamislio je MANJI od 50
 Unesi broj
 25
 Broj koji sam zamislio je VECI od 25
 Unesi broj
 30
 Broj koji sam zamislio je MANJI od 30
 Unesi broj
 27
 "BRAVO!!! Pogodio si! Zamislio sam 27. Bilo je lepo igrati se sa tobom. :)
```

[Rešenje 1.12]

Zadatak 1.13 **Aproksimacija broja PI metodom Monte Karlo** Napisati program koji aproksimira broj PI koriscenjem metode Monte Karlo. Sa standardnog ulaza unosi se broj N. Nakon toga N puta se bira tačka na slučajan način tako da su obe koordinate tačke iz intervala [0,1]. Broj PI se računa po sledecoj formuli:

$$PI = 4 * A/B$$

- A je broj slučajno izabranih tačaka koje pripadaju krugu poluprečnika 0.5, sa centrom u tački $(0.5, 0.5)$

- B je broj slučajno izabranih tačaka koje pripadaju kvadratu čija temena su tačke $(0, 0), (0, 1), (1, 1), (1, 0)$.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
POKRETANJE: aproksimacija_PI
 Izracunavanje broja PI metodom Monte Karlo
 Unesite broj iteracija:
 5
 Tacka:
 (0.14186247318019474, 0.15644650897353152)
 Tacka:
 (0.8910898038304426, 0.2200563958299553)
 Tacka:
 (0.641604107090444, 0.03712366524007682)
 Tacka:
 (0.4893727376942526, 0.17230005349431587)
 Tacka:
 (0.6199558112390107, 0.32122922953511124)
 Tacka:
 (0.5821041171248978, 0.025052299437462566)
 Broj PI aproksimiran metodom Monte Karlo: 4.0
```

[Rešenje 1.13]

Zadatak 1.14 **X-O** Napisati program koji implementira igricu X-O sa dva igrača.

Primer 1

```
Pokretanje: XO
Interakcija sa programom:
 IGRA: X-O pocinje
 Unesite ime prvog igraca:
 Ana
 Zdravo Ana
 Unesite ime drugog igraca:
 Petar
 Zdravo Petar!
 Igrac ('Ana', 'X') igra prvi.
 X : ('Ana', 'X')
 O : ('Petar', 'O')
 Zapocnimo igru
 TABLA
  1 2 3
  --- --- ---
 1 | - | - | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 1
 TABLA
  1 2 3
  --- --- ---
 1 | X | - | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Petar unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 2
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | - | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 2
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | - |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Petar unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 3
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | O |
  --- --- ---
 3 | - | - | - |
  --- --- ---
 Ana unesite koordinate polja koje
 zelite da popunite u posebnim linijama:
 Unesite vrstu:
 3
 Unesite kolonu:
 3
 TABLA
  1 2 3
  --- --- ---
 1 | X | O | - |
  --- --- ---
 2 | - | X | O |
  --- --- ---
 3 | - | - | X |
  --- --- ---
 BRAVO!!!!!!! Igrac Ana je pobedio!
```

[Rešenje 1.14]

1.1.3 Zadaci za vežbu

Zadatak 1.15 **Ajnc** Napisati program koji implementira igricu Ajnc sa jednim igračem. Igra se sa špilom od 52 karte. Na početku igrač unosi svoje ime nakon čega računar deli dve karte igraču i dve karte sebi. U svakoj sledećoj iteraciji računar deli po jednu kartu igraču i sebi. Cilj igre je sakupiti karte koje u zbiru imaju 21 poen. Karte sa brojevima nose onoliko bodova koliki je broj, dok žandar, dama, kralj nose 10 bodova. Karta As može da nosi 1 ili 10 bodova, u zavisnosti od toga kako igraču odgovara. Igrač koji sakupi 21 je pobedio. Ukoliko igrač premaši 21 bod, pobednik je njegov protivnik. Detaljan opis igre može se naći na narednoj adresi: https://en.wikipedia.org/wiki/Blackjack

Primer 1

```
Pokretanje: ajnc
Interakcija sa programom:
 -------- IGRA: Ajnc ---------
 Unesite Vase ime:
 Pavle
 Zdravo Pavle. :)
 Igra pocinje
 Vase karte su:
 1 Herc 5 karo
 Hit ili stand?[H/S]
 H
 Vase karte su:
 1 Herc 5 karo 5 tref
 Cestitam!!! Pobedio si!
 Bilo je lepo igrati se sa tobom. :)
```

Primer 2

```
Pokretanje: ajnc
Interakcija sa programom:
 -------- IGRA: Ajnc ---------
 Unesite Vase ime:
 Pavle
 Zdravo Pavle. :)
 Igra pocinje
 Vase karte su:
 Q Tref 7 karo
 Hit ili stand?[H/S]
 H
 Vase karte su:
 Q Tref 7 karo K Herc
 Zao mi je, izgubio si!:(
 Bilo je lepo igrati se sa tobom. :)
```

Zadatak 1.16 **4 u liniji** Napisati program koji implementira igricu 4 u nizu sa dva igrača. Tabla za igru je dimenzije 8x8. Igrači na početku unose svoja imena, nakon čega računar nasumično dodeljuje crvenu i žutu boju igračima. Igrač sa crvenom bojom igra prvi i bira kolonu u koju ce da spusti svoju lopticu. Cilj igre je da se sakupe 4 loptice iste boje u liniji. Prvi igrač koji sakupi 4 loptice u liniji je pobedio. Detaljan opis igre može se naći na narednoj adresi: https://en.wikipedia.org/wiki/Connect_Four

Primer 1

```
Pokretanje: cetri_u_nizu
Interakcija sa programom:
 IGRA: Cetiri u nizu pocinje
 Unesite ime prvog igraca:
 Ana
 Zdravo Ana
 Unesite ime drugog igraca:
 Petar
 Zdravo Petar!
 Igrac ('Ana', 'C') igra prvi.
 C : ('Ana', 'C')
 Z : ('Petar', 'Z')
 Zapocnimo igru
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
 Ana unesite koordinate polja
 koje zelite da popunite
 u posebnim linijama:
 Unesite vrstu:
 1
 Unesite kolonu:
 1
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | c | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
 Petar unesite koordinate polja
 koje zelite da popunite
 u posebnim linijama:
 Unesite vrstu:
 2
 Unesite kolonu:
 1
 TABLA
  1 2 3 4 5 6 7 8
  --- --- --- --- --- --- --- ---
 1 | c | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 2 | z | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 3 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 4 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 5 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 6 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 7 | - | - | - | - | - | - | - | - |
  --- --- --- --- --- --- --- ---
 8 | - | - | - | - | - | - | - | - |
```

```
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
2
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Petar unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
2
Unesite kolonu:
2
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
3
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
```

```
Petar unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
2
Unesite kolonu:
3
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | z | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
Ana unesite koordinate polja
koje zelite da popunite
u posebnim linijama:
Unesite vrstu:
1
Unesite kolonu:
4
TABLA
 1 2 3 4 5 6 7 8
 --- --- --- --- --- --- --- ---
1 | c | c | c | c | - | - | - | - |
 --- --- --- --- --- --- --- ---
2 | z | z | z | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
3 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
4 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
5 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
6 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
7 | - | - | - | - | - | - | - | - |
 --- --- --- --- --- --- --- ---
8 | - | - | - | - | - | - | - | - |
BRAVO!!!!!!! Igrac Ana je pobedio!
```

1.2 Datoteke, niske, JSON format, datum

1.2.1 Uvodni primeri

Zadatak 1.17 Korisnik na standardni ulaz unosi dve niske. Napisati program koji prvo pojavljivanje druge niske u prvoj zamenjuje karakterom $. U slučaju da nema pojavljivanja druge niske u prvoj i da je druga niska kraća ispisuje nadovezane niske sa separatorom -. Ako je druga niska duža od prve program treba da ispiše drugu nisku i njenu dužinu.

```
# Niske
#
# Mozemo ih pisati izmedju jednostrukih i dvostrukih navodnika

niska1 = raw_input("Unesite nisku: ")
niska2 = raw_input("Unesite nisku: ")

# Duzinu niske racunamo koristeci funkciju len(niska)
if len(niska1) > len(niska2):
    # Funkcija count
    # niska.count(podniska [, pocetak [, kraj]]) - vraca broj koliko se puta
    # podniska nalazi u niski (u intervalu od pocetak do kraj)
    n = niska1.count(niska2)
    if n > 0:
        # Funkcija find
        # niska.find(podniska [, pocetak [, kraj]]) - vraca poziciju prvog
    pojavljivanja
        # podniska u niski (u intervalu od pocetak do kraj), -1 ukoliko se podniska
    ne nalazi u niski
        i = niska1.find(niska2)
        # Karakterima u niski mozemo pristupati koristeci notaciju [] kao kod listi
        niska1 = niska1[0 : i] + '$' + niska1[i+len(niska2) : ]
        print niska1
    else:
        # Funkcija join
        # niska_separator.join([niska1,niska2,niska3,...]) - spaja listu niski
    separatorom
        print '-'.join([niska1,niska2])
else:
    print "Duznina niske {0:s} je {1:d}".format(niska2, len(niska2))

# Korisne funkcije za rad sa niskama:
#
# niska.isalnum()
#       isalpha()
#       isdigit()
#       islower()
#       isspace()
#       isupper()
# niska.split(separator) - razlaze nisku u listu koristeci separator
# niska.replace(stara, nova [, n]) - zamenjuje svako pojavljivanje niske stara
# niskom nova (ukoliko je zadat broj n, onda zamenjuje najvise n pojavljivanja)
```

Zadatak 1.18 Napisati program koji ispisuje tekući dan u nedelji, dan, mesec i vreme u formatu *hh:mm:ss*.

```
# Datumi
# Ukljucujemo klasu datetime iz modula datetime
from datetime import datetime

# Nov objekat datuma:
# datetime.datetime(godina, mesec, dan [, sat [, minut [, sekund]]])
# Korisne funkcije:
# datetime.now() - vraca trenutno vreme odnosno datum
# datetime.strptime(datum_niska, format)
# datetime.year, datetime.month, datetime.day, datetime.hour, datetime.minute,
    datetime.second,
# datetime.strftime(format) - vraca string reprezentaciju objekta datuma na osnovu
    zadatog formata
# datetime.strptime(niska, format) - vraca objekat datetime konstruisan na osnovu
    niske u zadatom formatu
```

```
# datetime.time([sat [, minut [, sekund]]]) - vraca objekat koji predstavlja vreme
# datetime.date(dan, mesec, godina) - vraca objekat datuma
# format:
#   %A - dan u nedelji (Monday, Tuesday,...)
#   %w - dan u nedelji (0, 1, 2,..., 6)
#   $d - dan (01, 02, 03,...)
#   %B - mesec (January, February,...)
#   %m - mesec (01, 02, ...)
#   %Y - godina (1992, 1993,...)
#   %H - sat (00, 01, ..., 23)
#   %M - minut (00, 01, ..., 59)
#   %S - sekund (00, 01, ..., 59)

print "\n-----Datumi-----\n"
print datetime.now().strftime("Dan u nedelji: %a/%w, Dan: %d, Mesec: %b/%m, Godina: %
    y, Vreme: %H:%M:%S\n")
print datetime.now().time()
print datetime.now().date()
```

Zadatak 1.19 Napisati program koji ispisuje sadrzaj datoteka *datoteka.txt* na standardni izlaz karakter po karakter.

```
# Datoteku otvaramo koristeci funkciju open koja vraca
# referencu na otvoreni tok podataka.
#
# rezimi:
# - "r"  -> read
# - "w"  -> write
# - "a"  -> append
# - "r+" -> read + append
#
# Datoteku smo duzni da zatvorimo sa 'datoteka.close()',
#
# datoteka = open("datoteka.txt","r")
# kod koji obradjuje datoteku
# ...
# datoteka.close()

# Python nudi 'with' koji omogucava da
# se datoteka automatski zatvori, cak i u situaciji
# kada se ispali izuzetak. Ovo je preporuceni nacin
# za citanje tokova podataka u Python-u.
with open("datoteka.txt", "r") as datoteka:
    # Citamo datoteku liniju po liniju, a potom
    # u liniji citamo karakter po karakter.
    for linija in datoteka:
        for karakter in linija:
            print karakter
    # datoteka.close() nam nije neophodno,
    # Python ce sam zatvoriti datoteku kada
    # se zavrsi 'with' blok
```

Zadatak 1.20 Napisati program koji ispisuje sadrzaj datoteka *datoteka.txt* na standardni izlaz liniju po liniju.

```
# Ispitivanje da li je otvaranje datoteke uspelo:
try:
    with open("datoteka.txt","r") as g:
        # Liniju po liniju mozemo ucitavati koristeci petlju
        # tako sto 'iteriramo' kroz Datoteku
        print "-----Iteriranje kroz datoteku <<for>> petljom-----\n"
        # Metod f.readline() cita jednu liniju iz Datoteke
        for linija in g:
            print linija
except IOError:
    # Ukoliko ne uspe otvaranje datoteke, Python ispaljuje
    # izuzetak 'IOError'.
    print "Nije uspelo otvaranje datoteke."
```

Zadatak 1.21 Napisati program koji dodaje u datoteku *datoteka.txt* nisku *water* a potom

ispisuje njen sadržaj na standardni izlaz.

```
# f.readlines() i list(f)
# vracaju listu linija datoteke
#
# f.write(niska) upisuje nisku u datoteku
print "-----Upisivanje u datoteku-----\n"
# TODO: razresiti konfuziju izmedju a+ i r+
with open("datoteka.txt","a+") as h:
    h.write("water\n")
    print h.readlines()
```

Zadatak 1.22 Korisnik na standarni ulaz unosi podatke o imenu, prezimenu i godinama. Program potom kreira JSON objekat *junak*, koji ima podatke *Ime*, *Prezime* i *Godine*, i ispisuje ga na standardni izlaz, a potom i u datoteku *datoteka.txt*.

```
# JSON format
#
# Funkcije za rad sa JSON formatom se nalaze u modulu json
import json

ime = raw_input("Unesite ime: ")
prezime = raw_input("Unesite prezime: ")
godine = int(raw_input("Unesite godine: "))

# json.dumps(objekat) vraca string koji sadrzi JSON reprezentaciju objekta x

print "\n-----JSON reprezentacija objekta-----\n"
junak = {"Ime": ime, "Prezime":prezime, "Godine":godine}
print json.dumps(junak)

# json.dump(x,f) upisuje string sa JSON reprezentacijom objekta x u datoteku f

f = open("datoteka.json","w")
json.dump(junak,f)
f.close()
```

Zadatak 1.23 Napisati program koji iz datoteke *datoteka.txt* učitava JSON objekat, a potom na standardni izlaz ispisuje podatke o *imenu*, *prezimenu* i *godininama*.

```
# json.load(f) ucitava iz datoteke string koji sadrzi
# JSON format objekta i vraca referencu na mapu koja
# je konstruisana na osnovu .json datoteke.
print "\n-----Ucitavanje objekta iz datoteke-----\n"
f = open("dat4.json","r")
x = json.load(f)
print x['Ime']
print x['Prezime']
print x['Godine']
f.close()
```

1.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.24 Napisati program koji sa standardnog ulaza učitava ime datoteke i broj n i računa broj pojavljivanja svakog n-grama u datoteci koji su sačinjeni od proizvoljnih karaktera i rezultat upisuje u datoteku `rezultat.json`.

Primer 1

```
Pokretanje: python n-anagram.py
Interakcija sa programom:
 Unesite ime datoteke:
 datoteka.txt
 Unesite n
 2
```

Sadržaj datoteka koje se koriste u primeru 1.24:

Listing 1.1: *datoteka.txt*

```
Ovo je datoteka dat
```

Listing 1.2: *rezultat.json*

```
{
'a ': 1, 'ka': 1, 'ot': 1, 'ek': 1,
' d': 2, ' j': 1, 'da': 2, 'e ': 1,
'o ': 1, 'to': 1, 'at': 2, 'je': 1,
'Ov': 1, 'te': 1, 'vo': 1
}
```

[Rešenje 1.24]

Zadatak 1.25

U datoteci `korpa.json` se nalazi spisak kupljenog voća u json formatu:

```
[ { 'ime' : ime_voca, 'kolicina' : broj_kilograma } , ...]
```

U datotekama `maxi_cene.json`, `idea_cene.json`, `shopngo_cene.json` se nalaze cene voća u json formatu:

```
[ { 'ime' : ime_voca, 'cena' : cena_po_kilogramu } , ...]
```

Napisati program koji izračunava ukupan račun korpe u svakoj prodavnici i ispisuje cene na standardni izlaz.

Primer 1

```
POKRETANJE: python korpa.py
INTERAKCIJA SA PROGRAMOM:
 Maxi: 631.67 dinara
 Idea: 575.67 dinara
 Shopngo: 674.67 dinara
```

[Rešenje 1.25]

Sadržaj datoteka koje se koriste u primeru 1.25:

Listing 1.3: *korpa.json*

```
[ {"ime" : "jabuke" ,"kolicina": 3.3},
{"ime": "kruske" , "kolicina": 2.1},
{"ime": "grozdje" , "kolicina":  2.6},
```

Listing 1.4: *maksi_cene.json*

```
[ {"ime" :"jabuke", "cena" : 59.9},
{"ime" :"kruske", "cena" : 120},
{"ime" :"grozdje", "cena" : 70},
{"ime" :"narandze", "cena" : 49.9},
{"ime" :"breskve", "cena" : 89.9} ]
```

Listing 1.5: *idea_cene.json*

```
[ {"ime" :"jabuke", "cena" : 39.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"breskve", "cena" : 59.9} ]
```

Listing 1.6: *shopngo_cene.json*

```
[ {"ime" :"jabuke", "cena" : 69.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"maline", "cena" : 290},
```

[Rešenje 1.25]

1.2.3 Zadaci za vežbu

Zadatak 1.26 Napisati program koji iz datoteke `ispiti.json` učitava podatke o ispitima i njihovim datumima. Ispisati na standardni izlaz za svaki ispit njegovo ime i status "Prosao" ukoliko je ispit prosao, odnosno "Ostalo je jos n dana.", gde je n broj dana od trenutnog datuma do datuma ispita.

Primer 1

```
POKRETANJE: python ispiti.py
INTERAKCIJA SA PROGRAMOM:
 Relacione baze podataka Prosao
 Vestacka inteligencija Prosao
 Linearna algebra i analiticka geometrija Prosao
```

Sadržaj datoteka koje se koriste u primeru 1.26:

Listing 1.7: *ispiti.json*

```
[  {'ime': 'Relacione baze podataka',
'datum': '21.09.2016.'},
{'ime': 'Vestacka inteligencija',
'datum': '17.06.2017.'},
{'ime': 'Linearna algebra i analiticka geometrija',
'datum': '08.02.2017.'} ]
```

Zadatak 1.27 Napisati program koji izdvaja sve jednolinijske i višelinijske komentare iz .c datoteke čije ime se unosi sa standardnog ulaza, listu jednih i drugih komentara upisuje u datoteku `komentari.json`. Jednolinijski komentari se navode nakon // a višelinijski između /∗ i ∗/.

Primer 1

```
POKRETANJE: python komentari.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 program.c
```

Sadržaj datoteka koje se koriste u primeru 1.27:

Listing 1.8: *program.c*

```
#include <stdio.h>

// Primer jednolinijskog komentara

int main(){
/*
Na ovaj nacin ispisujemo tekst
na standardni izlaz koristeci jezik C.
*/
printf("Hello world!");

// Na ovaj nacin se ispisuje novi red
printf("\n");
/*
Ukoliko se funkcija uspesno zavrsila
vracamo 0 kao njen rezultat.
*/
return 0;
}
```

Listing 1.9: *komentari.json*

```
{
  'jednolinijski' : ['Primer jednolinijskog komentara',
```

```
                      'Na ovaj nacin se ispisuje novi red'],
  'viselinijski' : ['Na ovaj nacin ispisujemo tekst na standardni
                      izlaz koristeci jezik C.',
                      'Ukoliko se funkcija uspesno zavrsila
                      vracamo 0 kao njen rezultat.']
}
```

Zadatak 1.28 Napisati program upoređuje dve datoteke čija imena se unose sa standardnog ulaza. Rezultat upoređivanja je datoteka `razlike.json` koja sadrži broj linija iz prve datoteke koje se ne nalaze u drugoj datoteci i obratno. *Napomena* Obratiti pažnju na efikasnost.

Primer 1

```
POKRETANJE: python razlika.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 dat1.txt
 Unesite ime datoteke:
 dat2.txt
```

Sadržaj datoteka koje se koriste u primeru 1.28:

Listing 1.10: *dat1.txt*

```
//netacno

same=1;

for(i=0;s1[i]!='\0' && s2[i]!='\0';i++) {
if(s1[i]!=s2[i]) {
same=0;
break;
}
}
return same;
```

Listing 1.11: *dat2.txt*

```
//tacno

for(i=0;s1[i]!='\0' && s2[i]!='\0';i++){

if(s1[i]!=s2[i])
return 0;
}
return s1[i]==s2[i];
```

Listing 1.12: *razlike.json*

```
{
  'dat1.txt' : 7,
  'dat2.txt' : 4
}
```

1.3 Argumenti komandne linije, sortiranje, obilazak direktorijuma

1.3.1 Uvodni primeri

Zadatak 1.29 Napisati program koji na standarni izlaz ispisuje argumente komandne linije.

```
# modul sys ima definisan objekat argv koji predstavlja listu argumenata komandne
    linije (svi argumenti se cuvaju kao niske karaktera)
```

```
2
import sys
4
if len(sys.argv) == 1:
6     print "Niste naveli argumente komandne linije"
      # funkcija exit() iz modula sys prekida program
8     # (ukoliko se ne prosledi argument, podrazumevano
      # se salje None objekat)
10    exit()

12 # ispisujemo argumente komandne linije
   # prvi argument, tj. sys.argv[0] je uvek ime skript fajla koji se pokrece
14 for item in sys.argv:
      print item
```

Zadatak 1.30 Napisati program koji na standrani izlaz ispisuje oznaku za tekući i roditeljski direktorijum, kao i separator koji se koristi za pravljenje putanje.

```
import os
2
print os.curdir     # oznaka tekuceg direktorijuma
4 print os.pardir    # oznaka za roditeljski direktorijum tekuceg direktorijuma
print os.sep        # ispisuje separator koji koristi za pravljenje putanja
```

Zadatak 1.31 Napisati program koji imitira rad komande *ls*. Program na standardni izlaz ispisuje sadržaj tekućeg direktorijuma.

```
1 import os

3 # funkcija za prosledjenu putanju direktorijuma vraca listu imena
  # svih fajlova u tom direktorijumu, . je zamena za putanju tekuceg direktorijuma
5 print os.listdir(os.curdir)

7 # os.walk() - vraca listu torki (trenutni_direktorijum, poddirektorijumi, datoteke)
  # os.path.join(putanja, ime) - pravi putanju tako sto nadovezuje na
9 #   prosledjenu putanju zadato ime odvojeno /

11 print "\n-----Prolazak kroz zadati direktorijum-----\n"
   for (trenutni_dir, poddirektorijumi, datoteke) in os.walk(os.curdir):
13     print trenutni_dir
       for datoteka in datoteke:
15         print os.path.join(trenutni_dir, datoteka)

17 # os.path.abspath(path) - vraca apsolutnu putanju za zadatu relativnu putanju nekog
       fajla
   # os.path.isdir(path) - vraca True ako je path putanja direktorijuma, inace vraca
       False
19 # os.path.isfile(path) - vraca True ako je path putanja regularnog fajla, inace vraca
        False
```

Zadatak 1.32 Napisati program koji na standarni izlaz ispisuje sve apsolutne putanje regularnih fajlova koji se nalaze u tekućem direktorijumu.

```
1 import os

3 print "\n-----Regularni fajlovi zadatog direktorijuma-----\n"
  for ime in os.listdir(os.curdir):
5     # Funkcija 'join' vrsi konkatenaciju putanja koristeci sistemski separator
      if os.path.isfile(os.path.join(os.curdir, ime)):
7         print os.path.abspath(os.path.join(os.curdir, ime))
```

Zadatak 1.33 U datoteci *tacke.json* se nalaze podaci o tačkama u sledećem formatu.

Listing 1.13: *tacke.json*

```
1 [ {"teme":"A" , "koordinate": [10.0, 1.1]},
2  {"teme":"B" , "koordinate": [1.0, 15.0]},
3  {"teme":"C" , "koordinate": [-1.0, 5.0]} ]
```

Napisati program koji učitava podatke o tačkama iz datoteke *tacke.json* i sortira i po udaljenosti od koordinatnog početka. Na standarni izlaz ispisati podatke pre i posle sortiranja.

```
# Sortiranje
#
# sorted(kolekcija [, poredi [, kljuc [, obrni]]]) - vraca sortiranu kolekciju
#
# kolekcija - kolekcija koju zelimo da sortiramo
# poredi - funkcija poredjenja
# kljuc - funkcija koja vraca kljuc po kome se poredi
# obrni - True/False (opadajuce/rastuce)
#
# za poziv sorted(kolekcija) koristi se funkcija cmp za poredjenje
# cmp(x, y) -> integer
# vraca negativnu vrednost za x<y, 0 za x==y, pozitivnu vrednost za x>y
# ako su x i y niske, cmp ih leksikografski poredi

import json
import math

# l = ["A", "C", "D", "5", "1", "3"]
# print l
# print "sortirana lista: ", sorted(l)

# u sledecem primeru je neophodno da definisemo svoje funkcije za poredjenje i
    vracanje kljuca jer je kolekcija lista recnika i za to cmp nema definisano
    ponasanje
with open("tacke.json","r") as f:
    tacke = json.load(f)

# funkcija koja tacke x i y poredi po njihovoj udaljenosti od koordinatnog pocetka
def poredi(x,y):
    if (x[0]*x[0] + x[1]*x[1]) > (y[0]*y[0] + y[1]*y[1]):
        return 1
    else:
        return -1
# funkcija kljuc kao argument ima element kolekcije koja se poredi, u ovom slucaju je
     to jedan recnik
# povratna vrednost funkcije kljuc je u stvari tip argumenata funkcije poredi
def kljuc(x):
    return x["koordinate"]

sortirane_tacke = sorted(tacke, poredi, kljuc) # ili sorted(tacke, poredi, kljuc,
    True) ako zelimo opadajuce da se sortira
print "Tacke pre sortiranja:"
for item in tacke:
        print item["teme"],
print "\nTacke nakon sortiranja: "
for item in sortirane_tacke:
        print item["teme"],
print
```

1.3.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.34 Napisati program koji računa odnos kardinalnosti skupova duže i šire za zadati direktorijum. Datoteka pripada skupu duže ukoliko ima više redova od maksimalnog broja karaktera po redu, u suprotnom pripada skupu šire. Sa standardnog ulaza se unosi putanja do direktorijuma. Potrebno je obići sve datoteke u zadatom direktorijumu i njegovim poddirektorijumima (koristiti funkciju os.walk()) i ispisati odnos kardinalnosti skupova duže i šire.

Primer 1

```
Pokretanje: python duze_sire.py
Interakcija sa programom:
 Unesite putanju do direktorijuma:
 ..
 Kardinalnost skupa duze : Kardinalnost skupa sire
 10 : 15
```

[Rešenje 1.34]

Zadatak 1.35 Napisati program koji obilazi direktorijume rekurzivno i računa broj datoteka za sve postojeće ekstenzije u tim direktorijumima. Sa standardnog ulaza se unosi putanja do početnog direktorijuma, a rezultat se ispisuje u datoteku `rezultat.json`.

Primer 1

```
POKRETANJE: python ekstenzije.py
INTERAKCIJA SA PROGRAMOM:
 Unesite putanju do direktorijuma:
 .
```

Sadržaj datoteka koje se koriste u primeru 1.35:

Listing 1.14: *rezultat.txt*

```
{
 'txt' : 14,
 'py' : 12,
 'c' : 10
}
```

[Rešenje 1.35]

Zadatak 1.36 U datoteci `radnici.json` nalaze se podaci o radnom vremenu zaposlenih u preduzeću u sledecem formatu:

```
[  { 'ime' : ime\_radnika, 'odmor' : [pocetak, kraj], 'radno_vreme'
: [pocetak, kraj] }, ...]
```

Napisati program koji u zavisnosti od unete opcije poslodavcu ispisuje trenutno dostupne radnike odnosno radnike koji su na odmoru. Moguće opcije su 'd' - trenutno dostupni radnici i 'o' - radnici koji su na odmoru. Radnik je dostupan ukoliko nije na odmoru i trenutno vreme je u okviru njegovog radnog vremena.

Primer 1

```
POKRETANJE: python odmor.py
INTERAKCIJA SA PROGRAMOM:
 "Unesite opciju koju zelite
 d - dostupni radnici
 o - radnici na odmoru :
 m
 Uneta opcija nije podrzana
```

Primer 2

```
POKRETANJE: python odmor.py
INTERAKCIJA SA PROGRAMOM:
 "Unesite opciju koju zelite
 d - dostupni radnici
 o - radnici na odmoru :
 d
 Pera Peric
```

[Rešenje 1.36]

Sadržaj datoteka koje se koriste u primeru 1.36:

Listing 1.15: *radnici.json*

```
[  { 'ime' : 'Pera Peric',
'odmor' : ['21.08.2016.', '31.08.2016.'],
'radno_vreme' : ['08:30', '15:30'] } ]
```

Zadatak 1.37 Napisati program koji učitava ime datoteke sa standardnog ulaza i na standardni izlaz ispisuje putanje do svih direktorijuma u kojima se nalazi ta datoteka.

Primer 1

```
POKRETANJE: python pojavljivanja.py
INTERAKCIJA SA PROGRAMOM:
 Unesite ime datoteke:
 1.py
 /home/student/vezbe/cas1/1.py
 /home/student/vezbe/cas7/1.py
 /home/student/vezbe/cas9/1.py
```

[Rešenje 1.39]

1.3.3 Zadaci za vežbu

Zadatak 1.38 Napisati program koji ispisuje na standardni izlaz putanje do lokacija svih Apache virtuelnih hostova na računaru. Smatrati da je neki direktorijum lokacija Apache virtuelnog hosta ukoliko u sebi sadrži `index.html` ili `index.php` datoteku.

Primer 1

```
POKRETANJE: python apache.py
INTERAKCIJA SA PROGRAMOM:
 /home/student/PVEB/prviPrimer
 /home/student/licna_strana
 /home/student/PVEB/ispit/jun
```

Zadatak 1.39 Napisati program koji realizuje autocomplete funkcionalnost. Sa standardnog ulaza korisnik unosi delove reči sve dok ne unese karakter !. Nakon svakog unetog dela reči ispisuju se reči koje počinju tim karakterima. Spisak reči koje program može da predloži se nalazi u datoteci `reci.txt`.

Primer 1

```
POKRETANJE: python autocomplete.py
INTERAKCIJA SA PROGRAMOM:
 ma
 mac macka mama maceta madjionicar
 mac
 mac macka maceta
 !
```

Sadržaj datoteka koje se koriste u primeru 1.39:

Listing 1.16: *reci.txt*

```
1 mac pesma skola macka mama maceta igra madjionicar
```

1.4 Rešenja

Rešenje 1.12 Pogodi broj

```
1  # Pogodi broj
2
3  import random
4
5  print "-------- IGRA: Pogodi broj ---------\n"
6
7  zamisljen_broj = random.randint(0,100)
8
9  ime = raw_input("Unesite Vase ime: ")
10
11 print "Zdravo {0:s}. :) \nZamislio sam neki broj od 1 do 100. Da li mozes da pogodis
       koji je to broj? ".format(ime)
12
13 pogodio = 0;
   while not pogodio:
15      print "Unesi broj:"
        broj = int(raw_input())
17      if broj == zamisljen_broj:
            pogodio = 1
19      elif broj > zamisljen_broj:
            print "Broj koji sam zamislio je MANJI od {0:d}.".format(broj)
21      else:
            print "Broj koji sam zamislio je VECI od {0:d}.".format(broj)
22
23 print "BRAVO!!! Pogodio si! Zamislio sam {0:d}. Bilo je lepo igrati se sa tobom. :)".
       format(zamisljen_broj)
```

Rešenje 1.13 Aproksimacija broja PI metodom Monte Karlo

```
# Aproksimacija broja PI metodom Monte Karlo

import random, math

def dist(A, B):
    """Funkcija izracunava euklidsko rastojanje izmedju tacaka A i B"""
    return math.sqrt((A[0]-B[0])**2 + (A[1]-B[1])**2)

print "Izracunavanje broja PI metodom Monte Karlo \n"
N = int(raw_input("Unesite broj iteracija: "))
A = 0   # Broj tacaka u krugu
B = 0   # Broj tacaka u kvadratu

i = N
while i >= 0:
    tacka = (random.random(), random.random())
    # Ukoliko se tacka nalazi u krugu, povecavamo broj tacaka u krugu
    if dist(tacka, (0.5, 0.5)) <= 0.5:
        A = A + 1
    B = B + 1
    i = i - 1

# Alternativno resenje:
# Generisemo N tacaka unutar kvadrata
# list comprehensions:
# http://www.pythonforbeginners.com/basics/list-comprehensions-in-python
xs = [(random.random(), random.random()) for x in range(N)]
# Izdvajamo tacke koje su unutar kvadrata
# lambde:
# https://pythonconquerstheuniverse.wordpress.com/2011/08/29/lambda_tutorial/
inside = filter(lambda (x, y): dist((0.5, 0.5), (x, y)) <= 0.5, xs)
A = len(inside)
B = N

print "Broj PI aproksimiran metodom Monte Karlo: "
print 4.0*A/B
```

Rešenje 1.14 X-O

```
# X-O
#
#  - | O | X
# --- --- ---
#  X | - | -
# --- --- ---
#  - | X | O

import random

def ispisi_tablu(tabla):
    print "\n    TABLA \n"
    print "    1   2   3   "
    print "   --- --- ---  "
    indeks = 1
    for i in tabla:
        print indeks,"|",i[0],"|",i[1],"|",i[2],"|"
        print "   --- --- --- "
        indeks = indeks + 1
    print "\n"

def pobedio(tabla):
    if (tabla[0][0] != "-" and tabla[0][2] != "-") and ((tabla[0][0] == tabla[1][1]
    == tabla[2][2]) or (tabla[0][2] == tabla[1][1] == tabla[2][0])):
        return True
    for i in range(3):
        if (tabla[0][i] != "-" and tabla[i][0] != "-") and ((tabla[0][i] == tabla[1][
    i] == tabla[2][i]) or (tabla[i][0] == tabla[i][1] == tabla[i][2])):
            return True
    return False

def ucitaj_koordinate(ime):
```

```
    while True:
        print "{0:s} unesite koordinate polja koje zelite da popunite u posebnim
    linijama:\n".format(ime)
        x = int(raw_input("Unesite vrstu: "))
        y = int(raw_input("Unesite kolonu: "))
        if 1<=x<=3 and 1<=y<=3:
            return x-1,y-1
        else:
            "Morate uneti brojeve 1,2 ili 3\n"

def korak(igrac):
    while True:
        x,y = ucitaj_koordinate(igrac[0])
        if tabla[x][y] == "-":
            tabla[x][y] = igrac[1]
            ispisi_tablu(tabla)
            break
        else:
            print tabla[x][y]
            print "Uneto polje je popunjeno!\n"

print "IGRA: X-O pocinje\n"

ime1 = raw_input("Unesite ime prvog igraca: ")
print "Zdravo {0:s}!\n".format(ime1)
ime2 = raw_input("Unesite ime drugog igraca: ")
print "Zdravo {0:s}!\n".format(ime2)

indikator = random.randint(1,2)
if indikator == 1:
    prvi_igrac = (ime1, "X")
    drugi_igrac = (ime2, "O")
else:
    prvi_igrac = (ime2, "X")
    drugi_igrac = (ime1, "O")

print "Igrac {0:s} igra prvi. \n".format(prvi_igrac)
print "X : {0:s}\n".format(prvi_igrac)
print "O : {0:s}\n".format(drugi_igrac)

tabla = [['-','-','-'],['-','-','-'],['-','-','-']]

print "Zapocnimo igru \n"

ispisi_tablu(tabla)

na_redu = 0
iteracija = 0
igraci = [prvi_igrac, drugi_igrac]
while iteracija < 9:
    korak(igraci[na_redu])
    if pobedio(tabla) == True:
        print "BRAVO!!!!!!! Igrac {0:s} je pobedio!\n".format(igraci[na_redu][0])
        break
    na_redu = (na_redu+1)%2
    iteracija = iteracija + 1

if iteracija == 9:
    print "NERESENO! Pokusajte ponovo.\n"
```

Rešenje 1.24

```
# dat.txt:
# Ovo je datoteka dat
#
# rezultat.json:
#
# {"a ": 1, "ka": 1, "ot": 1, "ek": 1, " d": 2, " j": 1, "da": 2, "e ": 1, "o ": 1, "
    to": 1, "at": 2, "je": 1, "Ov": 1, "te": 1, "vo": 1}

import json
```

```
ime_datoteke = raw_input("Unesite ime datoteke: ")
n = int(raw_input("Unesite broj n: "))

# Otvaramo datoteku i citamo njen sadrzaj
f = open(ime_datoteke, "r")
sadrzaj = f.read()
f.close()

recnik = {}
i = 0
# Prolazimo kroz sadrzaj i uzimamo jedan po jedan n-gram
while i < len(sadrzaj) - n:
    ngram = sadrzaj[i : i+n]
    # Ukoliko se n-gram vec nalazi u recniku,
    #  povecavamo mu broj pojavljivanja
    if ngram in recnik:
        recnik[ngram] = recnik[ngram]+1
    # Dodajemo n-gram u recnik i postavljamo mu broj na 1
    else:
        recnik[ngram] = 1
    i = i + 1

f = open("rezultat.json", "w")
json.dump(recnik,f)
f.close()
```

Rešenje 1.25

```
import json

def cena_voca(prodavnica, ime_voca):
  for voce in prodavnica:
    if voce['ime'] == ime_voca:
      return voce['cena']

# Ucitavamo podatke iz datoteka
f = open('korpa.json', "r")
korpa = json.load(f)
f.close()

f = open('maxi_cene.json', "r")
maxi_cene = json.load(f)
f.close()

f = open('idea_cene.json', "r")
idea_cene = json.load(f)
f.close()

f = open('shopngo_cene.json', "r")
shopngo_cene = json.load(f)
f.close()

maxi_racun = 0
idea_racun = 0
shopngo_racun = 0
i = 0
# Za svako voce u korpi dodajemo njegovu cenu u svaki racun posebno
while i < len(korpa):
    ime_voca = korpa[i]['ime']
    maxi_racun = maxi_racun + korpa[i]['kolicina']*cena_voca(maxi_cene, ime_voca)
    idea_racun = idea_racun + korpa[i]['kolicina']*cena_voca(idea_cene, ime_voca)
    shopngo_racun = shopngo_racun + korpa[i]['kolicina']*cena_voca(shopngo_cene,
    ime_voca)
    i += 1

print "Maxi: " + str(maxi_racun) + " dinara"
print "Idea: " + str(idea_racun) + " dinara"
print "Shopngo: " + str(shopngo_racun) + " dinara"
```

Rešenje 1.34

```
import os

dat_u_duze = 0
dat_u_sire = 0

# Funkcija koja obilazi datoteku i vraca 1 ukoliko datoteka pripada skupu duze
# odnosno 0 ukoliko datoteka pripada skupu sire
def obilazak(ime_datoteke):
    br_linija = 0
    najduza_linija = 0
    with open(ime_datoteke, "r") as f:
        for linija in f:
            br_linija = br_linija + 1
            if len(linija) > najduza_linija:
                najduza_linija = len(linija)
    if br_linija > najduza_linija:
        return 1
    else:
        return 0

ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")

for (tren_dir, pod_dir, datoteke ) in os.walk(ime_direktorijuma):
    for dat in datoteke:
        if obilazak(os.path.join(tren_dir, dat)) == 0:
            dat_u_sire += 1
        else:
            dat_u_duze += 1

print "Kardinalnost skupa duze: kardinalnost skupa sire"
print str(dat_u_duze)+":"+str(dat_u_sire)
```

Rešenje 1.35

```
import os
import json

ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")

ekstenzije = {}

for (tren_dir, pod_dir, datoteke) in os.walk(ime_direktorijuma):
    for dat in datoteke:
        pozicija = dat.find(".")
        # Ukoliko datoteka ima ekstenziju, pretpostavljamo da su datoteke imenovane
    tako da posle . ide ekstenzija u ispravnom obliku
        if pozicija >= 0:
            # Ukoliko ekstenzija postoji u mapi, povecavamo njen broj
            if dat[pozicija:] in ekstenzije:
                ekstenzije[dat[pozicija:]] += 1
            else:
            # Dodajemo novu ekstentiju u mapu i postavljamo njen broj na 1
                ekstenzije[dat[pozicija:]] = 1

with open("rezultat.json","w") as f:
    json.dump(ekstenzije, f)
```

Rešenje 1.36

```
import json, os, sys
from datetime import datetime

try:
    with open("radnici.json", "r") as f:
        radnici = json.load(f)
except IOError:
    print "Otvaranje datoteke nije uspelo!"
    sys.exit()
```

```
opcija = raw_input("Unesite opciju koju zelite (d - dostupni radnici, o - radnici na
    odmoru): \n")

if opcija != "d" and opcija != "o":
    print "Uneta opcija nije podrzana."
    exit()

tren_dat = datetime.now()

# funkcija datetime.strptime(string, format) pravi objekat tipa datetime na osnovu
    zadatih podataka u stringu i odgovarajuceg formata, na primer ako je datum
    zapisan kao "21.08.2016" odgovarajuci format je "%d.%m.%Y." pa se funkcija poziva
     sa datetime.strptime("21.08.2016", "%d.%m.%Y.")

for radnik in radnici:
    kraj_odmora = datetime.strptime(radnik['odmor'][1],"%d.%m.%Y.").date()
    pocetak_odmora = datetime.strptime(radnik['odmor'][0],"%d.%m.%Y.").date()
    kraj_rad_vrem = datetime.strptime(radnik['radno_vreme'][1],"%H:%M").time()
    pocetak_rad_vrem = datetime.strptime(radnik['radno_vreme'][0],"%H:%M").time()
    if opcija == "o":
        # Ukoliko je radnik trenutno na odmoru ispisujemo ga
        if  pocetak_odmora < tren_dat.date() < kraj_odmora:
            print radnik["ime"]
    else:
        # Ukoliko je radnik trenutno dostupan i nije na odmoru, ispisujemo ga
        if not (pocetak_odmora < tren_dat.date() < kraj_odmora) and pocetak_rad_vrem
    < tren_dat.time() < kraj_rad_vrem:
            print radnik["ime"]
```

Rešenje 1.39

```
import os

ime_datoteke = raw_input("Unesite ime datoteke: ")

# pretrazujemo ceo fajl sistem, odnosno pretragu krecemo od root direktorijuma /
# imajte u vidu da ce vreme izvrsavanja ovog programa biti veliko posto se pretrazuje
     ceo fajl sistem, mozete ga prekinuti u svakom trenutku sa CTRL+C
for (tren_dir, pod_dir, datoteke) in os.walk("/"):
    # objekat datoteke predstavlja listu imena datoteka iz direktorijuma
    # ta imena poredimo sa zadatim
    for dat in datoteke:
        # ako smo naisli na trazenu datoteku, pravimo odgovarajucu putanju
        if dat == ime_datoteke:
            print os.path.join(os.path.abspath(tren_dir), ime_datoteke)
```

2

Programiranje ograničenja - Python

Potrebno je imati instaliran Python 2.7 i biblioteku python-constraint. Na Ubuntu 14.04 operativnom sistemu, biblioteka python-constraint se moze instalirati pomocu Pip alata:

```
sudo apt-get -y install python-pip
sudo pip install python-constraint
```

Korisni linkovi i literatura:

http://labix.org/doc/constraint/
https://pypi.python.org/pypi/python-constraint
http://www.hakank.org/constraint_programming_blog/

2.1 Programiranje ograničenja

2.1.1 Uvodni primeri

Zadatak 2.1 Napisati program koji na standardni izlaz ispisuje sve kombinacije oblika xyz, gde je $x \in \{a, b, c\}$, $y \in \{1, 2, 3\}$ i $z \in \{0.1, 0.2, 0.3\}$ tako da važi da je $10 \cdot z = y$.

```
# Programiranje ogranicenja

# Ukljucujemo modul za rad sa ogranicenjima
import constraint

# Definisemo problem
problem = constraint.Problem()
# Dodajemo promenljive
#
# problem.addVariable(ime_promenljive, domen_lista)
# problem.addVariables(lista_imena_promenljivih, domen_lista)
problem.addVariable('x',['a','b','c'])
problem.addVariable('y',[1,2,3])
# Ispisujemo resenja
# print problem.getSolutions()

problem.addVariable('z',[0.1,0.2,0.3])
# Dodajemo ogranicenja
#
# problem.addConstraint(ogranicenje [, redosled_promenljivih])
#
# ogranicenje moze biti:
#   constraint.AllDifferentConstraint() - razlicite vrednosti svih promenljivih
#   constraint.AllEqualConstraint() - iste vrednosti svih promenljivih
#   constraint.MaxSumConstraint(s [,tezine]) - suma vrednosti promenljivih (pomnozena
     sa tezinama) ne prelazi s
#   constraint.MinSumConstraint(s [,tezine]) - suma vrednosti promenljivih (pomnozena
     sa tezinama) nije manja od s
#   constraint.ExactSumConstraint(s [,tezine]) - suma vrednosti promenljivih (
    pomnozena sa tezinama) je s
#   constraint.InSetConstraint(skup) - vrednosti promenljivih se nalaze u skupu skup
```

```
#   constraint.NotInSetConstraint(skup) - vrednosti promenljivih se ne nalaze u skupu
     skup
#   constraint.SomeInSetConstraint(skup) - vrednosti nekih promenljivih se nalaze u
    skupu skup
#   constraint.SomeNotInSetConstraint(skup) - vrednosti nekih promenljivih se ne
    nalaze u skupu skup
#
# redosled_promenljivih predstavlja listu promenljivih
# i zadaje se zbog definisanja tacnog redosleda
# ogranicenja koja se primenjuju na promenljive
#
# Mozemo napraviti i svoju funkciju ogranicenja
def ogranicenje(y,z):
    if y / 10.0 == z:
        return True

# Prosledjujemo funkciju ogranicenja i redosled promenljivih koji treba da odgovara
    redosledu argumenata funkcije ogranicenja
problem.addConstraint(ogranicenje,['y','z'])
resenja = problem.getSolutions()
print "\n-----Resenja-----\n"
for resenje in resenja:
    print str(resenje['x']) + " " + str(resenje['y'])+ " " + str(resenje['z'])
```

2.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 2.2 Napisati program koji pronalazi trocifren broj `ABC` tako da je količnik `ABC / (A + B + C)` minimalan i `A`, `B` i `C` su različiti brojevi.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print """
min_resenje['A']*100 + min_resenje['B']*10 + min_resenje['C']
"""
```

[Rešenje 2.2]

Zadatak 2.3 Dati su novčići od 1, 2, 5, 10, 20 dinara. Napisati program koji pronalazi sve moguće kombinacije tako da zbir svih novčića bude 50.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print """
1 din: {0:d}
2 din: {1:d}
5 din: {2:d}
10 din: {3:d}
20 din: {4:d}
""".format(r["1 din"],r["2 din"],r["5 din"], r["10 din"], r["20 din"])
```

[Rešenje 2.3]

Zadatak 2.4 Napisati program koji ređa brojeve u magičan kvadrat. Magičan kvadrat je kvadrat dimenzija 3x3 takav da je suma svih brojeva u svakom redu, svakoj koloni i svakoj dijagonali jednak 15 i svi brojevi različiti. Na primer:

```
4 9 2
3 5 7
8 1 6
```

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print " ------- "
print "| {0:d} {1:d} {2:d} |".format(r['a'],r['b'],r['c'])
print "| {0:d} {1:d} {2:d} |".format(r['d'],r['e'],r['f'])
print "| {0:d} {1:d} {2:d} |".format(r['g'],r['h'],r['i'])
print " ------- "
```

[Rešenje 2.4]

Zadatak 2.5 Napisati program koji pronalazi sve vrednosti promenljivih X, Y i Z za koje važi da je $X >= Z$ i $X*2+Y*X+Z <= 34$ pri čemu promenljive pripadaju narednim domenima $X \in \{1, 2, ..., 90\}$, $Y \in \{2, 4, 6, ...60\}$ i $Z \in \{1, 4, 9, 16, ..., 100\}$

[Rešenje 2.5]

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "-------------------------"
print "X = {0:d} , Y = {1:d} , Z = {2:d}".format(r['X'],r['Y'],r['Z'])
```

[Rešenje 2.5]

Zadatak 2.6 Napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
  TWO
 +TWO
 ---------
 FOUR
```

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "-----"
print "  " + str(r['T']) + str(r['W']) + str(r['O'])
print " +" + str(r['T']) + str(r['W']) + str(r['O'])
print "=" + str(r['F']) + str(r['O']) + str(r['U']) + str(r['R'])
```

[Rešenje 2.6]

Zadatak 2.7 Napisati program koji pronalazi sve vrednosti promenljivih X, Y, Z i W za koje važi da je $X >= 2 * W$, $3 + Y <= Z$ i $X - 11 * W + Y + 11 * Z <= 100$ pri čemu promenljive pripadaju narednim domenima $X \in \{1, 2, ..., 10\}$, $Y \in \{1, 3, 5, ...51\}$, $Z \in \{10, 20, 30, ..., 100\}$ i $W \in \{1, 8, 27, ..., 1000\}$.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "-------------------------"
print "X = {0:d} , Y = {1:d} , Z = {2:d}, W = {3:d}".format(r['X'],r['Y'],r['Z'], r['W'
    ])
```

[Rešenje 2.7]

Zadatak 2.8 Napisati program koji raspoređuje brojeve 1-9 u dve linije koje se seku u jednom broju. Svaka linija sadrži 5 brojeva takvih da je njihova suma u obe linije 25 i brojevi su u rastućem redosledu.

```
1   3
 2 4
  5
 6 8
7   9
```

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "_____"
print "{0:d}   {1:d}".format(r['a'],r['A'])
print " {0:d} {1:d} ".format(r['b'],r['B'])
print "  {0:d}  ".format(r['c'])
print " {0:d} {1:d} ".format(r['D'],r['d'])
print "{0:d}   {1:d}".format(r['E'],r['e'])
print "_____"
```

[Rešenje 2.8]

Zadatak 2.9 Pekara *Kiflica* proizvodi hleb i kifle. Za mešenje hleba potrebno je 10 minuta, dok je za kiflu potrebno 12 minuta. Vreme potrebno za pečenje ćemo zanemariti. Testo za hleb sadrži 300g brašna, a testo za kiflu sadrži 120g brašna. Zarada koja se ostvari prilikom prodaje jednog hleba je 7 dinara, a prilikom prodaje jedne kifle je 9 dinara. Ukoliko pekara ima 20 radnih sati za mešenje peciva i 20kg brašna, koliko komada hleba i kifli treba da se umesi kako bi se ostvarila maksimalna zarada (pod pretpostavkom da će pekara sve prodati)?

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "_____"
print "Maksimalna zarada je {0:d}, komada hleba je {1:d}, a komada kifli {2:d}".format
    (7*max_H + 9*max_K, max_H, max_K)
print "_____"
```

[Rešenje 2.9]

Zadatak 2.10 Napisati program pronalazi vrednosti A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S (svako slovo predstavlja različit broj) koje su poređane u heksagon na sledeći način:

```
  A,B,C
 D,E,F,G
H,I,J,K,L
 M,N,O,P
  Q,R,S
```

tako da zbir vrednosti duž svake horizontalne i dijagonalne linije bude 38 (A+B+C = D+E+F+G = ... = Q+R+S = 38, A+D+H = B+E+I+M = ... = L+P+S = 38, C+G+L = B+F+K+P = ... = H+M+Q = 38).

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print " _____"
print "  {0:d},{1:d},{2:d}".format(r['A'],r['B'],r['C'])
print " {0:d},{1:d},{2:d},{3:d}".format(r['D'],r['E'],r['F'],r['G'])
print "{0:d},{1:d},{2:d},{3:d},{4:d}".format(r['H'],r['I'],r['J'],r['K'],r['L'])
print " {0:d},{1:d},{2:d},{3:d}".format(r['M'],r['N'],r['O'],r['P'])
print "  {0:d},{1:d},{2:d}".format(r['Q'],r['R'],r['S'])
print " _____"
```

[Rešenje 2.10]

Zadatak 2.11 Kompanija Start ima 250 zaposlenih radnika. Rukovodstvo kompanije je odlučilo da svojim radnicima obezbedi dodatnu edukaciju. Da bi se radnik obučio programskom jeziku Elixir potrebno je platiti 100 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 150 projekat/sati mesečno, što bi za kompaniju značilo dobit od 5 evra po projekat/satu. Da bi se radnik obučio programskom jeziku Dart potrebno je platiti 105 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 170 projekat/sati mesečno, koji bi za kompaniju značili dobit od 6 evra po satu. Ukoliko Start ima na raspolaganju 26000 evra za obuku i maksimalan broj 51200 mogućih projekat/sati mesečno, odrediti na koji nacin kompanija treba da obuči svoje zaposlene kako bi ostvarila maksimalnu dobit.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print """Maksimalna zarada je {0:d},
broj radnika koje treba poslati na kurs Elixir je {1:d},
a broj radnika koje treba poslati na kurs Dart je {2:d}.
""".format(170*6*max_E + 150*5*max_D - (100*max_E + 150*max_D)  , max_E, max_D)
```

[Rešenje 2.11]

Zadatak 2.12 Napisati program koji raspoređuje 8 topova na šahovsku tablu tako da se nikoja dva topa ne napadaju.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "-----------------"
for i in "12345678":
  for j in range(1,9):
    if r[i] == j:
      print "T",
    else:
      print "-",
  print ""
print "-----------------"
```

[Rešenje 2.12]

Zadatak 2.13 Napisati program koji raspoređuje 8 dama na šahovsku tablu tako da se nikoje dve dame ne napadaju.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "-----------------"
for i in "12345678":
  for j in range(1,9):
    if r[i] == j:
      print "D",
    else:
      print "-",
  print ""
print "-----------------"
```

[Rešenje 2.13]

Zadatak 2.14 Napisati program koji učitava tablu za Sudoku iz datoteke čije ime se zadaje sa standardnog ulaza i korišćenjem ograničenja rešava Sudoku zagonetku.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "=========================="
for i in range(1,10):
  print "|",
  for j in range(1,10):
    if j%3 == 0:
      print str(r[i*10+j])+" |",
    else:
      print str(r[i*10+j]),
  print ""
  if i%3 == 0 and i!=9:
    print "--------------------------"
print "=========================="
```

Primer 1

```
Pokretanje: python sudoku.py
Interakcija sa programom:
 Unesite ime datoteke sa tablom za sudoku:
 sudoku.json
 ========================
 | 6 3 8 | 7 5 2 | 1 9 4 |
 | 9 2 7 | 4 1 8 | 5 6 3 |
 | 5 4 1 | 6 3 9 | 2 7 8 |
 ------------------------
 | 7 1 4 | 5 9 6 | 3 8 2 |
 | 2 8 9 | 1 7 3 | 4 5 6 |
 | 3 5 6 | 8 2 4 | 7 1 9 |
 ------------------------
 | 8 9 5 | 3 4 1 | 6 2 7 |
 | 4 7 2 | 9 6 5 | 8 3 1 |
 | 1 6 3 | 2 8 7 | 9 4 5 |
 ========================
```

[Rešenje 2.14]

Sadržaj datoteke koja se koriste u primeru 2.14:

Listing 2.1: *sudoku.json*

```
[[6, 3, 8, 7, 0, 0, 0, 0, 0],
[0, 2, 7, 0, 0, 0, 5, 0, 0],
[5, 4, 0, 6, 0, 9, 2, 0, 0],
[0, 1, 0, 5, 9, 0, 3, 0, 0],
[0, 0, 9, 0, 7, 0, 4, 0, 0],
[0, 0, 6, 0, 2, 4, 0, 1, 0],
[0, 0, 5, 3, 0, 1, 0, 2, 7],
[0, 0, 2, 0, 0, 0, 8, 3, 0],
[0, 0, 0, 0, 0, 7, 9, 4, 5]]
```

2.1.3 Zadaci za vežbu

Zadatak 2.15 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
GREEN + ORANGE = COLORS
MANET + MATISSE + MIRO + MONET + RENOIR = ARTISTS
COMPLEX + LAPLACE = CALCULUS
THIS + IS + VERY = EASY
CROSS + ROADS = DANGER
FATHER + MOTHER = PARENT
WE + WANT + NO + NEW + ATOMIC = WEAPON
EARTH + AIR + FIRE + WATER = NATURE
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS
SEE + YOU = SOON
NO + GUN + NO = HUNT
WHEN + IN + ROME + BE + A = ROMAN
DONT + STOP + THE = DANCE
HERE + THEY + GO = AGAIN
OSAKA + HAIKU + SUSHI = JAPAN
MACHU + PICCHU = INDIAN
SHE + KNOWS + HOW + IT = WORKS
COPY + PASTE + SAVE = TOOLS
```

Sve rezultate ispisati na standardni izlaz koristeći sledeći format komande ispisa. Primer za prvu zagonetku.

Komanda ispisa rešenja:

```
print "   " + str(r['G']) + str(r['R']) + str(r['E']) + str(r['E']) + str(r['N'])
```

```
print " +" + str(r['O']) + str(r['R']) + str(r['A']) + str(r['N']) + str(r['G']) + str(r
    ['E'])
print "=" + str(r['C']) + str(r['O']) + str(r['L']) + str(r['O']) + str(r['R']) + str(r[
    'S'])
```

Zadatak 2.16 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
THREE + THREE + ONE = SEVEN
NINE + LESS + TWO = SEVEN
ONE + THREE + FOUR = EIGHT
THREE + THREE + TWO + TWO + ONE = ELEVEN
SIX + SIX + SIX = NINE + NINE
SEVEN + SEVEN + SIX = TWENTY
ONE + ONE + ONE + THREE + THREE + ELEVEN = TWENTY
EIGHT + EIGHT + TWO + ONE + ONE = TWENTY
ELEVEN + NINE + FIVE + FIVE = THIRTY
NINE + SEVEN + SEVEN + SEVEN = THIRTY
TEN + SEVEN + SEVEN + SEVEN + FOUR + FOUR + ONE = FORTY
TEN + TEN + NINE + EIGHT + THREE = FORTY
FOURTEEN + TEN + TEN + SEVEN = FORTYONE
NINETEEN + THIRTEEN + THREE + TWO + TWO + ONE + ONE + ONE = FORTYTWO
FORTY + TEN + TEN = SIXTY
SIXTEEN + TWENTY + TWENTY + TEN + TWO + TWO = SEVENTY
SIXTEEN + TWELVE + TWELVE + TWELVE + NINE + NINE = SEVENTY
TWENTY + TWENTY + THIRTY = SEVENTY
FIFTY + EIGHT + EIGHT + TEN + TWO + TWO = EIGHTY
FIVE + FIVE + TEN + TEN + TEN + TEN + THIRTY = EIGHTY
SIXTY + EIGHT + THREE + NINE + TEN = NINETY
ONE + NINE + TWENTY + THIRTY + THIRTY = NINETY
```

Sve rezultate ispisati na standardni izlaz koristeći sledeći format komande ispisa. Primer za prvu zagonetku.

Komanda ispisa rešenja:

```
print "   " + str(r['T']) + str(r['R']) + str(r['E']) + str(r['E'])
print "   " + str(r['T']) + str(r['R']) + str(r['E']) + str(r['E'])
print " +" + str(r['O']) + str(r['N']) + str(r['E'])
print "=" + str(r['S']) + str(r['E']) + str(r['V']) + str(r['E']) + str(r['N'])
```

Zadatak 2.17 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da jednakost bude zadovoljena:

```
MEN * AND = WOMEN
COGITO = ERGO * SUM
((JE + PENSE) - DONC) + JE = SUIS
FERMAT * S = LAST + THEOREM.
WINNIE / THE = POOH
TWO * TWO + EIGHT = TWELVE
```

Sve rezultate ispisati na standardni izlaz koristeći sledeći format komande ispisa. Primer za prvu zagonetku.

Komanda ispisa rešenja:

```
print "   " + str(r['M']) + str(r['A']) + str(r['N'])
print " *" + str(r['A']) + str(r['N']) + str(r['D'])
print "=" + str(r['W']) + str(r['O']) + str(r['M']) + str(r['E']) + str(r['N'])
```

Zadatak 2.18 Uraditi sve zadatke koji su pobrojani ovde:
http://www.primepuzzle.com/leeslatest/alphameticpuzzles.html

Zadatak 2.19 Napisati program koji učitava ceo broj n i ispisuje magičnu sekvencu S brojeva od 0 do $n-1$. $S = (x_0, x_1, ..., x_{n-1})$ je magična sekvenca ukoliko postoji x_i pojavljivanja broja i za $i = 0, 1, ..., n-1$.

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "( ",
for i in range(n-1):
  print str(r["x"+str(i)]) + ", ",
print str(r["xn-1"]) + ")"
```

Primer 1

```
Pokretanje: python sudoku.py
Interakcija sa programom:
 Unesite dužinu magične sekvence
 4
 (1,2,1,0)
 (2,0,2,0)
```

Zadatak 2.20 Čistačica Mica sređuje i čisti kuće i stanove. Da bi sredila i počistila jedan stan potrebno joj je 1 sat, dok joj je za kuću potrebno 1.5 sati. Prilikom čišćenja, Mica potroši neku količinu deterdženta, 120ml po stanu, odnosno 100ml po kući. Mica zaradi 1000 dinara po svakom stanu, odnosno 1500 dinara po kući. Ukoliko Mica radi 40 sati nedeljno i ima 5l deterdženta na raspolaganju, koliko stanova i kuća je potrebno da očisti kako bi imala najveću zaradu?

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "Potrebno je da ocisti {0:d} kuca i {1:d} stanova da bi zarada bila najveca".
    format(r["kuca"], r["stan"])
```

Zadatak 2.21 Marija se bavi grnčarstvom i pravi šolje i tanjire. Da bi se napravila šolja, potrebno je 6 minuta, dok je za tanjir potrebno 3 minuta. Pri pravljenju šolje potroši se 75 gr, dok se za tanjir potroši 100 gr gline. Ukoliko ima 20 sati na raspolaganju za izradu svih proizvoda i 250 kg gline, a zarada koju ostvari iznosi 2 evra po svakoj šolji i 1.5 evra po tanjiru, koliko šolja i tanjira treba da napravi kako bi ostvarila maksimalnu zaradu?

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "Potrebno je da napravi {0:d} solja i {1:d} tanjira da bi zarada bila najveca".
    format(r["solja"], r["tanjir"])
```

Zadatak 2.22 Jovanin komšija preprodaje računare i računarsku opremu. Očekuje isporuku računara i štampača. Pri tom, računari su spakovani tako da njihova kutija zauzima 360 kubnih decimetara prostora, dok se štampači pakuju u kutijama koje zauzimaju 240 kubnih decimetara prostora. Komšija se trudi da mesečno proda najmanje 30 računara i da taj broj bude bar za 50% veći od broja prodatih štampača. Računari koštaju 200 evra po nabavnoj ceni, a prodaju se po ceni od 400 evra, dok štampači koštaju u nabavci 60 evra i prodaju se za 140 evra. Magacin kojim komšija raspolaže ima svega 30000 kubnih decimetara prostora i mesečno može da nabavi robu u iznosu od najviše 14000 evra. Koliko računara, a koliko štampača komšija treba da proda kako bi se maksimalno obogatio?

Sve rezultate ispisati na standardni izlaz koristeći datu komandu ispisa.

Komanda ispisa rešenja:

```
print "Potrebno je da proda {0:d} stampaca i {1:d} racunara da bi se maksimalno obogatio
    ".format(r["stampac"], r["racunar"])
```

2.2 Rešenja

Rešenje 2.2

```
1
import constraint
3
problem = constraint.Problem()
5 # Definisemo promenljive i njihove vrednosti
problem.addVariable('A',range(1,10))
7 problem.addVariable('B',range(10))
problem.addVariable('C',range(10))
9 # Dodajemo ogranicenje da su vrednosti svih promenljivih razlicite
problem.addConstraint(constraint.AllDifferentConstraint())
11 resenja = problem.getSolutions()
# Znamo da minimalni kolicnik mora biti manji od 999
13 min_kolicnik = 999
min_resenje = {}
15 for resenje in resenja:
    a = resenje['A']
17     b = resenje['B']
    c = resenje['C']
19     kolicnik = (float)(a*100 + b*10 + c) / (a+b+c)
    if kolicnik < min_kolicnik:
21         min_kolicnik = kolicnik
        min_resenje = resenje
23 print min_resenje['A']*100 + min_resenje['B']*10 + min_resenje['C']
```

Rešenje 2.3

```
1 import constraint

3 problem = constraint.Problem()
# Definisemo promenljive za svaki novcic
5 # ako bi se zahtevalo da u kombinaciji bude od svake vrednosti
# bar po jedan novcic samo treba promeniti da domen za svaku
7 # promenljivu krece od 1
problem.addVariable("1 din",range(0,51))
9 problem.addVariable("2 din",range(0,26))
problem.addVariable("5 din",range(0,11))
11 problem.addVariable("10 din",range(0,6))
problem.addVariable("20 din",range(0,3))
13
# Problem koji je uocen pri ispisu resenja je sledeci,
15 # redosled u kom ce biti dodate promenljive problemu ne
# mora uvek da odgovara redosledu kojim smo mi definisali promenljive,
17 # u konkretnom primeru (videti oblik u kom ispisuje resenje),
# promenljive ce se dodati u sledecem redosledu:
19 # '1 din', '2 din', '10 din', '20 din', '5 din'
# (nacin na koji se kljucevi organizuju u recniku nije striktno definisan,
21 # primetimo da niske nisu sortirane)
# posledica je da postavljanje ogranicenja
23 # problem.addConstraint(constraint.ExactSumConstraint(50,[1,2,5,10,20]))
# nece ispravno dodeliti tezine, na primer,
25 # tezinu 5 dodeli promenljivoj '10 din' umesto '5 din' kako bismo ocekivali

27 # I nacin da se resi ovaj problem je da redosled promenljivih
# koji odgovara redosledu tezina za ExactSumConstraint prosledimo
29 # kao dodatni argument za funkciju addConstraint

31 problem.addConstraint(
    constraint.ExactSumConstraint(50,[1,2,5,10,20]),
33     ["1 din", "2 din", "5 din","10 din", "20 din"])

35 # II nacin je da definisemo svoju funkciju koja predstavlja ogranicenje, samo ce sada
     solver nesto sporije da radi posto ugradjene funkcije imaju optimizovanu
    pretragu i brze dolaze do resenja
#
37 #def o(a, b, c, d, e):
# if a + 2*b + 5*c + 10*d + 20*e == 50:
```

```
#   return True
#
#problem.addConstraint(o, ["1 din", "2 din", "5 din","10 din", "20 din"])
#
resenja = problem.getSolutions()

for r in resenja:
  print """1 din: {0:d}
2 din: {1:d}
5 din: {2:d}
10 din: {3:d}
20 din: {4:d}
""".format(r["1 din"],r["2 din"],r["5 din"], r["10 din"], r["20 din"])

  # Provera da je suma bas 50
  print r["1 din"] + r["2 din"]*2 + r["5 din"]*5 + r["10 din"]*10 + r["20 din"]*20
```

Rešenje 2.4

```
#  4 9 2
#  3 5 7
#  8 1 6
#

import constraint

def o(x,y,z):
    if x+y+z == 15:
        return True

problem = constraint.Problem()
# Promenljive:
# a b c
# d e f
# g h i
problem.addVariables("abcdefghi", range(1,10))
problem.addConstraint(constraint.AllDifferentConstraint())
# Dodajemo ogranicenja za svaku vrstu
problem.addConstraint(o,"abc")
problem.addConstraint(o,"def")
problem.addConstraint(o,"ghi")
# Dodajemo ogranicenja za svaku kolonu
problem.addConstraint(o,"adg")
problem.addConstraint(o,"beh")
problem.addConstraint(o,"cfi")
#Dodajemo ogranicenja za dijagonale
problem.addConstraint(o,"aei")
problem.addConstraint(o,"ceg")

resenja = problem.getSolutions()
for r in resenja:
    print " ------- "
    print "| {0:d} {1:d} {2:d} |".format(r['a'],r['b'],r['c'])
    print "| {0:d} {1:d} {2:d} |".format(r['d'],r['e'],r['f'])
    print "| {0:d} {1:d} {2:d} |".format(r['g'],r['h'],r['i'])
    print " ------- "
```

Rešenje 2.5

```
# X,Y,Z
#
# X >= Z
# X*2 + X*Y + Z <= 34
#
# X <- {1,2,3,...90}
# Y <- {2,4,6,...60}
# Z <- {1,4,9,16,...100}
#
#
```

```
import constraint

problem = constraint.Problem()

# Dodajemo promenljivu X i definisemo njen domen
problem.addVariable('X', range(1,91))

# Dodajemo promenljivu Y i definisemo njen domen
problem.addVariable('Y', range(2,61,2))

domenZ = [];
for i in range(1,11):
    domenZ.append(i*i)

# Dodajemo promenljivu Z i definisemo njen domen
problem.addVariable('Z', domenZ)

def o1(x,z):
    if x >= z:
        return True

def o2(x,y,z):
    if x*2 + x*y + z <= 34:
        return True;

# Dodajemo ogranicenja
problem.addConstraint(o1, 'XZ')
problem.addConstraint(o2, 'XYZ')

resenja = problem.getSolutions()

for r in resenja:
    print "-------------------------"
    print "X = {0:d} , Y = {1:d} , Z = {2:d}".format(r['X'],r['Y'],r['Z'])
print "-------------------------"
```

Rešenje 2.6

```
#   TWO
#  +TWO
#  ------
#  FOUR
#

import constraint

problem = constraint.Problem()
# Definisemo promenljive i njihove vrednosti
problem.addVariables("TF",range(1,10))
problem.addVariables("WOUR",range(10))

# Definisemo ogranicenje za cifre
def o(t, w, o, f, u, r):
    if 2*(t*100 + w*10 + o) == f*1000 + o*100 + u*10 + r:
        return True

# Dodajemo ogranicenja za cifre
problem.addConstraint(o,"TWOFUR")
# Dodajemo ogranicenje da su sve cifre razlicite
problem.addConstraint(constraint.AllDifferentConstraint())

resenja = problem.getSolutions()

for r in resenja:
    print "-----"
    print "  "+str(r['T'])+str(r['W'])+str(r['O'])
    print " +"+str(r['T'])+str(r['W'])+str(r['O'])
    print "="+str(r['F'])+str(r['O'])+str(r['U'])+str(r['R'])
```

Rešenje 2.7

```
# Dati sistem nejednacina nema resenje, tj. metog getSolutions() vraca praznu listu
# Ukoliko se za promenljivu W domen promeni na {1,...,100} sistem ce imati resenje
import constraint

problem = constraint.Problem()

# Dodajemo promenljivu X i definisemo njen domen
problem.addVariable('X', range(1,11))

# Dodajemo promenljivu Y i definisemo njen domen
problem.addVariable('Y', range(1,52,2))

domenZ = []
domenW = []

for i in range(1,11):
  domenZ.append(i*10)
  domenW.append(i**3)

# Dodajemo promenljivu Z i definisemo njen domen
problem.addVariable('Z', domenZ)

# Dodajemo promenljivu W i definisemo njen domen
problem.addVariable('W', domenW)

# Za ovako definisan domen za promenljivu W sistem ce imati resenje
# problem.addVariable('W', range(1,101))


def o1(x,w):
    if x >= 2*w:
        return True

def o2(y,z):
    if 3 + y <= z:
        return True

def o3(x,y,z,w):
    if x - 11*w + y + 11*z <= 100:
        return True;

# Dodajemo ogranicenja
problem.addConstraint(o1, 'XW')
problem.addConstraint(o2, 'YZ')
problem.addConstraint(o3, 'XYZW')

resenja = problem.getSolutions()
# Proveravamo da li postoji resenje za sistem nejednacina
if resenja==[]:
  print "Sistem nema resenje."
else:
  for r in resenja:
        print "------------------------"
print "X = {0:d} , Y = {1:d} , Z = {2:d}, W = {3:d}".format(r['X'],r['Y'],r['Z'], r['
    W'])
    print "------------------------"
```

Rešenje 2.8

```
#       1   3
#        2 4
#         5
#        6 8
#       7   9
#

import constraint

# Definisemo ogranicenje za jednu dijagonalu
def o(a,b,c,d,e):
    if a<b<c<d<e and a+b+c+d+e==25:
```

```
        return True

problem = constraint.Problem()
# Definisemo promenljive za svaku poziciju
problem.addVariables('abcdeABDE',range(1,10))
# Dodajemo ogranicenja za obe dijagonale
problem.addConstraint(o,'abcde')
problem.addConstraint(o,'ABcDE')
# Dodajemo ogranicenje da su vrednosti svih promenljivih razlicite
problem.addConstraint(constraint.AllDifferentConstraint())

resenja = problem.getSolutions()
for r in resenja:
    print "-------------"
    print "{0:d}   {1:d}".format(r['a'],r['A'])
    print " {0:d} {1:d} ".format(r['b'],r['B'])
    print "  {0:d}  ".format(r['c'])
    print " {0:d} {1:d} ".format(r['D'],r['d'])
    print "{0:d}   {1:d}".format(r['E'],r['e'])
    print "-------------"
```

Rešenje 2.9

```
#
# Potrebno je napraviti H komada hleba i K komada kifli
#
# Zarada iznosi:
# - 7din/hleb, tj. zarada za H komada hleba bice 7*H
# - 9din/kifla tj. zarada za K komada kifli bice 9*K
#
# Ukupna zarada iznosi:
# 7*H + 9*K - funkcija koju treba maksimizovati
#
# Ogranicenja vremena:
# - vreme potrebno za mesenje jednog hleba je 10min,
#   tj. za mesenje H komada hleba potrebno je 10*H minuta
# - vreme potrebno za mesenje jedne kifle je 12min,
#   tj. za mesenje K komada kifli potrebno je 12*K minuta
#
# Ukupno vreme koje je na raspolaganju iznosi 20h, tako da je:
# 10*H + 12*K <= 1200
#
# Ogranicenje materijala:
# - za jedan hleb potrebno je 300g brasna, a za H komada hleba potrebno je H*300
    grama
# - za jednu kifli potrebno je 120g brasna, a za K komada kifli potrebno je K*120
    grama
#
# Ukupno, na raspolaganju je 20kg brasna, tako da je:
# 300*H + 120*K <= 20000
#
# Broj kifli i hleba je najmanje 0, tako da:
# H>=0
# K>=0
#
# S obzirom na to da imamo 20kg brasna na raspolaganju, mozemo napraviti:
# - najvise 20000/120 kifli
# - najvise 20000/300 hleba
#
# H <= 20000/120 ~ 167
# K <= 20000/300 ~ 67
#
# S obzirom na to da imamo 20h na raspolaganju, mozemo napraviti:
# - najvise 1200/12 kifli
# - najvise 1200/10 hleba
#
# H <= 1200/10 = 120
# K <= 1200/12 = 100
#
# najoptimalnije je za gornju granicu domena postaviti
# minimum od dobijenih vrednosti,
# tj. sve ukupno H <= 120, K <= 67
```

```
import constraint

problem = constraint.Problem()

# Dodajemo promenljivu H i definisemo njen domen
problem.addVariable('H', range(0,121))

# Dodajemo promenljivu K i definisemo njen domen
problem.addVariable('K', range(0,68))

def ogranicenje_vremena(h,k):
    if 10*h + 12*k <= 1200:
        return True

def ogranicenje_materijala(h,k):
    if 300*h + 120*k <= 20000:
        return True;

# Dodajemo ogranicenja vremena i matrijala
problem.addConstraint(ogranicenje_vremena, 'HK')
problem.addConstraint(ogranicenje_materijala, 'HK')

resenja = problem.getSolutions()

# Pronalazimo maksimalnu vrednost funkcije cilja
max_H = 0
max_K = 0

for r in resenja:
    if 7*r['H'] + 9*r['K'] > 7*max_H + 9*max_K:
        max_H = r['H']
        max_K = r['K']

print "----------------------------------------------------------------"
print "Maksimalna zarada je {0:d}, komada hleba je {1:d}, a komada kifli {2:d}".
    format(7*max_H + 9*max_K, max_H, max_K)
print "----------------------------------------------------------------"
```

Rešenje 2.10

```

import constraint

def o1(x,y,z):
    if x+y+z == 38:
        return True
def o2(x,y,z,w):
    if x+y+z+w == 38:
        return True
def o3(x,y,z,w,h):
    if x+y+z+w+h == 38:
        return True

problem = constraint.Problem()
problem.addVariables("ABCDEFGHIJKLMNOPQRST", range(1,38))
problem.addConstraint(constraint.AllDifferentConstraint())
# Dodajemo ogranicenja za svaku horizontalnu liniju
#  A,B,C
# D,E,F,G
#H,I,J,K,L
# M,N,O,P
#  Q,R,S
problem.addConstraint(o1,"ABC")
problem.addConstraint(o2,"DEFG")
problem.addConstraint(o3,"HIJKL")
problem.addConstraint(o2,"MNOP")
problem.addConstraint(o1,"QRS")

# Dodajemo ogranicenja za svaku od glavnih dijagonala
#  A,B,C
# D,E,F,G
```

```
#H,I,J,K,L
# M,N,O,P
#  Q,R,S

problem.addConstraint(o1,"HMQ")
problem.addConstraint(o2,"DINR")
problem.addConstraint(o3,"AEJOS")
problem.addConstraint(o2,"BFKP")
problem.addConstraint(o1,"CGL")

# Dodajemo ogranicenja za svaku od sporednih dijagonala
#  A,B,C
# D,E,F,G
#H,I,J,K,L
# M,N,O,P
#  Q,R,S

problem.addConstraint(o1,"ADH")
problem.addConstraint(o2,"BEIM")
problem.addConstraint(o3,"CFJNQ")
problem.addConstraint(o2,"GKOR")
problem.addConstraint(o1,"LPS")

resenja = problem.getSolutions()
for r in resenja:
    print " -----------------"
    print "  {0:d},{1:d},{2:d}".format(r['A'],r['B'],r['C'])
    print " {0:d},{1:d},{2:d},{3:d}".format(r['D'],r['E'],r['F'],r['G'])
    print "{0:d},{1:d},{2:d},{3:d},{4:d}".format(r['H'],r['I'],r['J'],r['K'],r['L'])
    print " {0:d},{1:d},{2:d},{3:d}".format(r['M'],r['N'],r['O'],r['P'])
    print "  {0:d},{1:d},{2:d}".format(r['Q'],r['R'],r['S'])
    print " -----------------"
```

Rešenje 2.11

```
import constraint

problem = constraint.Problem()
# Kompanija ima 250 zaposlenih radnika
# za sve njih organizuje dodatnu obuku
# ako je E promenjiva za Elixir, a D za Dart
# mora da vazi E<=250, D<=250 i E + D = 250
#
# Dodajemo promenljivu E i definisemo njen domen
problem.addVariable('E', range(0,251))

# Dodajemo promenljivu D i definisemo njen domen
problem.addVariable('D', range(0,251))

def ukupno_radnika(e,d):
  if e+d == 250:
    return True

def ogranicenje_projekat_sati(e,d):
    if 150*e + 170*d <= 51200:
        return True

def ogranicenje_sredstava(e,d):
    if 100*e + 105*d <= 26000:
        return True;

# Dodajemo ogranicenja za broj projekat/sati i ukupna sredstva
# na raspolaganju kao i za broj radnika u firmi
problem.addConstraint(ogranicenje_projekat_sati, 'ED')
problem.addConstraint(ogranicenje_sredstava, 'ED')
problem.addConstraint(ukupno_radnika, 'ED')

resenja = problem.getSolutions()

# Pronalazimo maksimalnu vrednost funkcije cilja
max_E = 0
max_D = 0
```

```
# Od ostvarene dobiti preko broja projekat/sati oduzimamo gubitak za placanje kurseva
    radnicima
for r in resenja:
    if 150*5*r['E'] + 170*6*r['D'] - (100*r['E'] + 105*r['D']) > 150*5*max_E + 170*6*
    max_D - (100*max_E + 105*max_D) :
        max_E = r['E']
        max_D = r['D']


print "Maksimalna zarada je {0:d}, broj radnika koje treba poslati na kurs Elixir je
    {1:d}, a broj radnika koje treba poslati na kurs Dart je {2:d}.".format(170*6*
    max_E + 150*5*max_D - (100*max_E + 150*max_D) , max_E, max_D)
```

Rešenje 2.12

```
# Jedan od mogucih rasporeda
# sva ostala resenja su permutacije
# takve da je u svakoj vrsti i koloni samo jedan top
#
# 8 T - - - - - - -
# 7 - T - - - - - -
# 6 - - T - - - - -
# 5 - - - T - - - -
# 4 - - - - T - - -
# 3 - - - - - T - -
# 2 - - - - - - T -
# 1 - - - - - - - T
#   1 2 3 4 5 6 7 8
#

import constraint

problem = constraint.Problem()
# Dodajemo promenljive za svaku kolonu i njihove vrednosti 1-8
problem.addVariables("12345678", range(1,9))
# Dodajemo ogranicenje da se topovi ne napadaju medjusobno po svakoj vrsti
problem.addConstraint(constraint.AllDifferentConstraint())
resenja = problem.getSolutions()

# Broj svih mogucih permutacija je 8! = 40320
# Za prikaz svih najbolje pozvati program sa preusmerenjem izlaznih podataka
# python 2_12.py > izlaz.txt
print "Broj resenja je: {0:d}.".format(len(resenja))

for r in resenja:
    print "---------------"
    for i in "12345678":
        for j in range(1,9):
            if r[i] == j:
                print "T",
            else:
                print "-",
        print ""
    print "---------------"
```

Rešenje 2.13

```
# 8 D - - - - - - -
# 7 - - - - D - - -
# 6 - D - - - - - -
# 5 - - - - - D - -
# 4 - - D - - - - -
# 3 - - - - - - D -
# 2 - - - D - - - -
# 1 - - - - - - - D
#   1 2 3 4 5 6 7 8
#

import constraint
import math
```

```
problem = constraint.Problem()
# Dodajemo promenljive za svaku kolonu i njihove vrednosti 1-8
problem.addVariables("12345678", range(1,9))
# Dodajemo ogranicenje da se dame ne napadaju medjusobno po svakoj vrsti
problem.addConstraint(constraint.AllDifferentConstraint())

for k1 in range(1,9):
    for k2 in range(1,9):
        if k1 < k2:
            # Definisemo funkciju ogranicenja za dijagonale
            def o(vrsta1, vrsta2, kolona1=k1, kolona2=k2):
                if math.fabs(vrsta1 - vrsta2) != math.fabs(kolona1 - kolona2):
                    return True
            problem.addConstraint(o, [str(k1),str(k2)])

resenja = problem.getSolutions()
# Za prikaz svih najbolje pozvati program sa preusmerenjem izlaznih podataka
# python 2_13.py > izlaz.txt
print "Broj resenja je: {0:d}.".format(len(resenja))
for r in resenja:
    print "---------------"
    for i in "12345678":
        for j in range(1,9):
            if r[i] == j:
                print "D",
            else:
                print "-",
        print ""
    print "---------------"
```

Rešenje 2.14

```
# 9 - - - - - - - - -
# 8 - - - - - - - - -
# 7 - - - - - - - - -
# 6 - - - - - - - - -
# 5 - - - - - - - - -
# 4 - - - - - - - - -
# 3 - - - - - - - - -
# 2 - - - - - - - - -
# 1 - - - - - - - - -
#   1 2 3 4 5 6 7 8 9
#

import constraint
import json

problem = constraint.Problem()

# Dodajemo promenljive za svaki red i njihove vrednosti 1-9
for i in range(1, 10):
    problem.addVariables(range(i * 10 + 1, i * 10 + 10), range(1, 10))

# Dodajemo ogranicenja da se u svakoj vrsti nalaze razlicite vrednosti
for i in range(1, 10):
    problem.addConstraint(constraint.AllDifferentConstraint(), range(i * 10 + 1, i *
    10 + 10))

# Dodajemo ogranicenja da se u svakoj koloni nalaze razlicite vrednosti
for i in range(1, 10):
    problem.addConstraint(constraint.AllDifferentConstraint(), range(10 + i, 100 + i,
     10))

# Dodajemo ogranicenja da svaki podkvadrat od 3x3 promenljive ima razlicite vrednosti
for i in [1,4,7]:
    for j in [1,4,7]:
        pozicije = [10*i+j,10*i+j+1,10*i+j+2,10*(i+1)+j,10*(i+1)+j+1,10*(i+1)+j
    +2,10*(i+2)+j,10*(i+2)+j+1,10*(i+2)+j+2]
        problem.addConstraint(constraint.AllDifferentConstraint(),pozicije)

```

```
ime_datoteke = raw_input("Unesite ime datoteke sa tablom za sudoku: ")
f = open(ime_datoteke, "r")
tabla = json.load(f)
f.close()

# Dodajemo ogranicenja za svaki broj koji je zadat na tabli
for i in range(9):
    for j in range(9):
        if tabla[i][j] != 0:
            def o(vrednost_promenljive, vrednost_na_tabli = tabla[i][j]):
                if vrednost_promenljive == vrednost_na_tabli:
                    return True

            problem.addConstraint(o, [((i+1)*10 + (j+1))])

resenja = problem.getSolutions()

for r in resenja:
    print "=========================="
    for i in range(1,10):
        print "|",
        for j in range(1,10):
            if j%3 == 0:
                print str(r[i*10+j])+" |",
            else:
                print str(r[i*10+j]),
        print ""
        if i%3 == 0 and i!=9:
            print "--------------------------"
    print "=========================="
```

3

Funkcionalno programiranje

Potrebno je imati instaliran GHC kompajler na računaru.
Literatura:

(a) https://www.haskell.org/

(b) https://wiki.haskell.org/Haskell

3.1 Uvod

3.1.1 Uvodni primeri

Zadatak 3.1 Napisati funkciju `main` koja ispisuje poruku `Zdravo! :)`.

```
-- Ovako pisemo jednolinijske komentare

{-
    Ovako pisemo
    viselinijske
    komentare
-}

{-
  Interpreter pokrecemo iz terminala komandom:

  ghci

  i otvorice nam se interaktivni interpreter:

  GHCi, version 8.0.1: http://www.haskell.org/ghc/  :? for help
  Prelude>

  Sa interpreterom mozemo direktno komunicirati, npr
  Prelude> print "Zdravo! :)"
  "Zdravo! :)"

  Interpretatoru dodajemo mogucnost izvrsavanja funkcija iz izvorne
  ime_programa.hs komandom:

  :load ime_programa.hs

  Nakon toga mozemo pokrenuti sve funkcije koje su u toj datoteci
  definisane, npr.

  Prelude> main
  "Zdravo! :)"

  Iz interpretera se izlazi komandom :quit
  Prelude> :quit

  Haskell programe mozemo kompajlirati komandom:

  ghc ime_programa.hs
```

```
  koja ce napraviti izvrsni program koji pokrecemo sa

  ./ime_programa

-}

main = print "Zdravo! :)"
```

Zadatak 3.2 Napisati funkciju `duplo n` koja računa dvostruku vrednost celog broja `n`.

```
{-
  Osnovni tipovi:
  Bool
  Char
  String
  Int
  Integer
  Float
  Double

  Tipski razredi (definisu funkcije koje neki tip mora da implementira):
  Eq - tipovi sa jednakoscu (==,/=)
  Ord - tipovi sa uredjenjem (<,<=,>,>=,...)
  Num - numericki tipovi (+,-,*,...)
  Integral - celobrojni tipovi (div, mod,...)
  Fractional - razlomacki tipovi (/, recip,...)

  Funkcije preslikavaju vrednosti jednog tipa (ili tipskog razreda) u vrednost drugog
     tipa (ili tipskog razreda):

  duplo :: Int -> Int

  Informacije o bilo kom objektu mozete dobiti naredbom:
  :info ime_objekta
-}

duplo :: Int -> Int
duplo n = n + n
```

Zadatak 3.3 Napisati funkciju `ostatak3 n` koja računa ostatak pri deljenju broja `n` brojem tri.

```
{-
  Aritmeticke operacije nad tipovima Int, Integer, Float:
  +, -, *, /,
  ^^ - stepen (tip Int)
  ** - stepen (tip Float)

  Funkcije:
  mod, div, log, sqrt, sin, cos, tan

-}

ostatak3 n = mod n 3
 -- ekvivalentno sa x `mod` 3 ukoliko koristimo infiksnu notaciju
 -- :info ime_funkcije - na kraju svih informacija je navedena infiksna notacija za
    trazenu funkciju
```

Zadatak 3.4 Napisati funkciju `korenCeli n` koja računa realni koren celog broja `n` korišćenjem ugrađene funkcije `sqrt`.

```
-- ukoliko koristimo funkcije koje su definisane za realne tipove nad argumentima
    koji su celobrojnog tipa potrebno je prevesti vrednost u nadklasu Num koriscenjem
     funkcije fromIntegral, nece se izvrsiti eksplicitno kastovanje
-- funkcija racuna vrednost korena samo za celobrojni argument

korenCeli :: Int -> Double
korenCeli n = sqrt (fromIntegral n)
```

Zadatak 3.5 Napisati funkciju `sumaPrvih n` koja računa sumu prvih `n` prirodnih brojeva (rekurzivno, bez korišćenja formule).

```
-- Zadatak se moze resiti koriscenjem uslovnog izraza ili ogradjenih jednacina

-- Blokovi se u Haskelu odvajaju tabulatorom
-- Uslovni izraz (mora imati else granu)

sumaPrvih n = if n == 0
          then 0
          else n + sumaPrvih(n-1)

-- Ogradjene jednacine (guarded equations) - mozemo ih koristiti umesto uslovnih
    izraza, bitan je redosled navodjenja, otherwise slucaj treba uvek staviti na
    kraju

{- sumaPrvih n
     | n == 0 = 0
     | otherwise = n + sumaPrvih(n-1)
-}
```

Zadatak 3.6 Napisati funkciju `lista a b` koja pravi listu celih brojeva iz segmenta $[a, b]$. U slučaju da granice segmenta nisu ispravne, rezultat je prazna lista.

```
{-
  Lista je niz vrednosti istog tipa:

  [element1,element2,...]

  Operacije:

  x : lista - dodaje element x na pocetak liste
  lista1 ++ lista1 - nadovezuje dve liste
  lista !! pozicija - vraca element liste koji se nalazi na poziciji pozicija
  [a..b] - konstruise listu sa elementima od a do b (u zavisnosti od tipa)

  Korisne funkcije:
  head lista - vraca prvi element liste
  tail lista - vraca rep liste
  length lista - vraca broj elemenata liste
  take n lista - vraca listu prvih n elemenata
  drop n lista - vraca listu bez prvih n elemenata
  null lista - vraca True ukoliko je lista prazna, False inace


  Niske su liste karaktera, na primer:

  niska1 = ['A','l','a','d','d','i','n']
  niska2 = "Aladdin"

-}

lista :: Int->Int->[Int]
lista a b = [a..b]
```

Zadatak 3.7 Napisati funkciju `parMax p` koja određuje veći element iz para realnih brojeva p.

```
{-
  N-torke su nizovi vrednosti razlicitog ili istog tipa:
  ('a',1,[1.2, 2.3],"torka")

  Funkcije definisane za parove (n=2):
  fst () - vraca prvi element
  snd () - vraca drugi element

-}

parMax :: (Double, Double) -> Double
parMax p | fst p <= snd p = snd p
         | otherwise = fst p
```

3.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 3.8 Napisati funkciju `proizvodPrvih n` koja računa proizvod prvih `n` prirodnih brojeva (rekurzivno, bez korišćenja formule). Pretpostaviti da je argument ispravan.

[Rešenje 3.8]

Primer 1

```
POKRETANJE: proizvodPrvih 5
IZLAZ:
 120
```

Primer 2

```
POKRETANJE: proizvodPrvih 30
IZLAZ:
 265252859812191058636308480000000
```

Zadatak 3.9 Napisati funkciju `prost n` koja vraća `True` ako je `n` prost, `False` inače. Pretpostaviti da je argument ispravan.

[Rešenje 3.9]

Primer 1

```
POKRETANJE: prost 5
IZLAZ:
 True
```

Primer 2

```
POKRETANJE: prost 12
IZLAZ:
 False
```

Zadatak 3.10 Napisati funkciju `nzd a b` koja računa najveći zajednički delilac brojeva `a` i `b` (koristiti Euklidov algoritam). Pretpostaviti da su argumenti ispravni.

[Rešenje 3.10]

Primer 1

```
POKRETANJE: nzd 12 15
IZLAZ:
 3
```

Primer 2

```
POKRETANJE: nzd 12 13
IZLAZ:
 1
```

Zadatak 3.11 Napisati funkciju `tipJednacine a b c` koja vraća tip kvadratne jednačine $a * x^2 + b * x + c = 0$ (`Degenerisana`, `Jedno resenje`, `Dva resenja`, `Bez resenja`).

[Rešenje 3.11]

Primer 1

```
POKRETANJE: tipJednacine 1 2 1
IZLAZ:
 "Jedno resenje"
```

Primer 2

```
POKRETANJE: tipJednacine (-1) 8 5
IZLAZ:
 "Bez resenja"
```

Zadatak 3.12 Napisati funkciju `izDekadne x osn` koja prebacuje broj `x` iz dekadne u osnovu `osn` i funkciju `uDekadnu x osn` koja prebacuje broj `x` iz osnove `osn` u dekadnu osnovu. Pretpostaviti da je $osn > 1$ i $osn < 10$.

[Rešenje 3.12]

Primer 1

```
POKRETANJE: izDekadne 101 2
IZLAZ:
 1100101
```

Primer 2

```
POKRETANJE: uDekadnu 765 8
IZLAZ:
 501
```

Zadatak 3.13 Napisati funkciju `ceoDeo x` koja računa ceo deo korena pozitivnog broja `x` (bez korišćenja ugrađenih funkcija za koren i/ili stepen).

[Rešenje 3.13]

Primer 1

```
POKRETANJE: ceoDeo 15
IZLAZ:
 3
```

Primer 2

```
POKRETANJE: ceoDeo 100
IZLAZ:
 10
```

Zadatak 3.14 Napisati funkciju `harm n` koja pravi listu prvih `n` elemenata harmonijskog reda. Pretpostaviti da je argument ispravan.

[Rešenje 3.14]

Primer 1

```
Pokretanje: harm 2
Izlaz:
 [1.0,0.5]
```

Primer 2

```
Pokretanje: harm 5
Izlaz:
 [1.0,0.5,0.3333333333333333,0.25,0.2]
```

Zadatak 3.15 Napisati funkciju `delioci n` koja pravi listu svih pravih delioca pozitivnog broja `n`. Pretpostaviti da je argument ispravan.

[Rešenje 3.15]

Primer 1

```
Pokretanje: delioci 12
Izlaz:
 [2,3,4,6]
```

Primer 2

```
Pokretanje: delioci 13
Izlaz:
 []
```

Zadatak 3.16 Napisati funkciju `nadovezi l1 l2 n` koja nadovezuje na listu `l1` `n` puta listu `l2`. Pretpostaviti da je argument `n` pozitivan broj.

[Rešenje 3.16]

Primer 1

```
Pokretanje: nadovezi [1] [2] 3
Izlaz:
 [1,2,2,2]
```

Primer 2

```
Pokretanje: nadovezi [] [1,2,3] 2
Izlaz:
 [1,2,3,1,2,3]
```

3.1.3 Zadaci za vežbu

Zadatak 3.17 Napisati funkciju `sumaKvadrata n` koja računa sumu kvadrata prvih `n` prirodnih brojeva (rekurzivno, bez korišćenja formule).

Primer 1

```
Pokretanje: sumaKvadrata 5
Izlaz:
 55
```

Primer 2

```
Pokretanje: sumaKvadrata 10
Izlaz:
 385
```

Zadatak 3.18 Napisati funkciju `brojDelilaca n` koja vraća broj pravih delilaca prirodnog broja `n`.

Primer 1

```
Pokretanje: brojDelilaca 5
Izlaz:
 0
```

Primer 2

```
Pokretanje: brojDelilaca 12
Izlaz:
 4
```

Zadatak 3.19 Napisati funkciju `fib n` koja računa `n`-ti element Fibonačijevog niza. Pretpostaviti da je argument ispravan.

Primer 1

```
Pokretanje: fib 5
Izlaz:
 5
```

Primer 2

```
Pokretanje: fib 12
Izlaz:
 144
```

Zadatak 3.20 Napisati funkciju `osnova x osn1 osn2` koja prebacuje broj `x` iz osnove `osn1` u osnovu `osn2`. Pretpostaviti da su `osn1` i `osn1` brojevi veći od 1 i manji od 10.

Primer 1

```
Pokretanje: osnova 100 10 3
Izlaz:
 10201
```

Primer 2

```
Pokretanje: osnova 525 8 2
Izlaz:
 101010101
```

Zadatak 3.21 Napisati funkciju `parni n` koja pravi listu prvih `n` parnih prirodnih brojeva. Pretpostaviti da je argument ispravan.

Primer 1

```
Pokretanje: parni 3
Izlaz:
 [2,4,6]
```

Primer 2

```
Pokretanje: parni 10
Izlaz:
 [2,4,6,8,10,12,14,16,18,20]
```

Zadatak 3.22 Napisati funkciju `fibLista n` koja pravi listu prvih `n` elemenata Fibonačijevog niza. Pretpostaviti da je argument ispravan.

Primer 1

```
Pokretanje: fibLista 5
Izlaz:
 [1,1,2,3,5]
```

Primer 2

```
Pokretanje: fibLista 10
Izlaz:
 [1,1,2,3,5,8,13,21,34,55]
```

Zadatak 3.23 Napisati funkciju `jednocifreniDelioci n` koja pravi listu svih jednocifrenih delilaca prirodnog broja `n`. Pretpostaviti da je argument ispravan.

Primer 1

```
Pokretanje: jednocifreniDelioci 15
Izlaz:
 [1,3,5]
```

Primer 2

```
Pokretanje: jednocifreniDelioci 40
Izlaz:
 [1,2,4,5,8]
```

3.2 Liste

3.2.1 Uvodni primeri

Zadatak 3.24 Napisati funkciju koja računa dužinu proizvoljne liste bez i sa korišćenjem šablona liste.

```
{-
  Uparivanje sablona (pattern matching)

  not :: Bool -> Bool
  not True = False
  not False = True

  Ako pozovemo not True uparice se prvi sablon,
  a ako pozovemo not False uparice se drugi sablon

  Dzoker (wildcard)

  prvi nacin (ako bismo zamenili redosled sablona, javlja pattern match overlapped)
  (&&) :: Bool -> Bool -> Bool
  True && True = True
  _ && _ = False   --- moglo bi i: n && m = False, bitno je samo da su razlicito
    imenovani

  drugi nacin, efikasniji:
  (&&) :: Bool -> Bool -> Bool
  True && x = x
  False && _ = False

  _ predstavlja dzoker, tj. sablon koji odgovara bilo kojoj vrednosti (obratiti
    paznju na redosled definisanja sablona!)

  Sabloni lista:

  length [_] = 1 - [_] predstavlja listu sa jednim elementom
  length x = 1 + length (tail x) - x predstavlja listu

-}

length1 [] = 0
length1 x = 1 + length1 (tail x)

-- (x:xs) - predstavlja sablon liste, x je glava, xs je rep
```

```
37 length2 [] = 0
   length2 (_:xs) = 1 + length2 xs
```

Zadatak 3.25 Napisati funkcije koje određuju glavu i rep proizvoljne liste bez korišćenja ugrađenih funkcija za rad sa listama.

```
-- preslikava iz liste elemenata u jedan element
glava :: [a] -> a
glava (x:_) = x

-- preslikava iz liste u listu
rep :: [a] -> [a]
rep (_:xs) = xs
```

Zadatak 3.26 Napisati funkciju `parni a b` koja generiše listu parnih celih brojeva iz segmenta $[a, b]$ i funkciju `neparni a b` koja generiše listu neparnih celih brojeva iz segmenta $[a, b]$.

```
-- koristimo generatore liste
-- even proverava da li je argument paran, a odd da li je neparan
-- elementi listi pripadaju skupu od 1 do 50
parni a b = [x | x<-[a..b], even x]
neparni a b = [x | x<-[a..b], odd x]
```

Zadatak 3.27 Napisati funkciju `parovi a b c d` koja generiše listu parova celih brojeva (x, y), za koje `x` pripada segmentu $[a, b]$, a `y` pripada segmentu $[c, d]$.

```
-- Mozemo imati vise generatora

parovi a b c d = [(x,y) | x<-[a..b], y<-[c..d]]

{- obratiti paznju kako utice zamena mesta generatora
parovi a b c d = [(x,y) | y<-[c..d], x<-[a..b]]
-}
```

Zadatak 3.28 Napisati funkciju `zavisnoY a b` koja generiše listu parova celih brojeva (x, y), za koje `x` pripada segmentu $[a, b]$, a `y` pripada segmentu $[x, b]$.

```
-- Generatori mogu zavisiti jedni od drugih

-- kasniji generatori mogu zavisiti samo od promenljivih koje su uvedene ranijim
    generatorima citajuci sa leva na desno
zavisnoY a b = [(x,y) | x<-[a..b], y<-[x..b]]
```

3.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 3.29 Napisati funkciju `bezbedanRep l` koja ukoliko je lista `l` prazna vraća praznu listu, inače vraća rep liste `l`, koristeći: a) uslovne izraze b) ograđene jednačine c) uparivanje šablona

[Rešenje 3.29]

Primer 1

```
Pokretanje: bezbedanRep [1,2,3]
Izlaz:
  [2,3]
```

Primer 2

```
Pokretanje: bezbedanRep []
Izlaz:
  []
```

Zadatak 3.30 Napisati funkciju `savrseni n` koja pravi listu savršenih brojeva manjih od `n`. Broj je savršen ukoliko je jednak sumi svojih faktora (tj. delilaca), ne uključujući taj broj.

[Rešenje 3.30]

Primer 1

```
POKRETANJE: savrseni 50
IZLAZ:
 [6,28]
```

Primer 2

```
POKRETANJE: savrseni 1000
IZLAZ:
 [6,28,496]
```

Zadatak 3.31 Napisati funkciju `zbirPar n` koja pravi listu parova (a, b) takvih da su `a` i `b` prirodni brojevi čiji je zbir jednak `n`.

[Rešenje 3.31]

Primer 1

```
POKRETANJE: zbirPar 1
IZLAZ:
 []
```

Primer 2

```
POKRETANJE: zbirPar 10
IZLAZ:
 [(1,9),(2,8),(3,7),(4,6),(5,5),(6,4),(7,3),(8,2),(9,1)]
```

Zadatak 3.32 Napisati funkciju `poslednji l` koja određuje poslednji element proizvoljne liste `l`.

[Rešenje 3.32]

Primer 1

```
POKRETANJE: poslednji ["ponedeljak","nedelja"]
IZLAZ:
 "nedelja"
```

Primer 2

```
POKRETANJE: poslednji [5,4,3,2,1]
IZLAZ:
 1
```

Zadatak 3.33 Napisati funkciju `spoji l` koja spaja listu listi istog tipa `l` u jednu listu.

[Rešenje 3.33]

Primer 1

```
POKRETANJE: spoji [["jedan"],["tri"],["pet"]]
IZLAZ:
 ["jedan","tri","pet"]
```

Primer 2

```
POKRETANJE: spoji [[],[1],[1,2],[1,2,3]]
IZLAZ:
 [1,1,2,1,2,3]
```

Zadatak 3.34 Napisati funkciju `sufiksi l` koja pravi listu svih sufiksa proizvoljne liste `l`.

[Rešenje 3.34]

Primer 1

```
POKRETANJE: sufiksi [1,2,3]
IZLAZ:
 [[1,2,3],[2,3],[3],[]]
```

Primer 2

```
POKRETANJE: sufiksi []
IZLAZ:
 [[]]
```

Zadatak 3.35 Napisati funkciju `izbaci k l` koja izbacuje `k`-ti element iz liste `l`. U slučaju da je zadata neispravna pozicija u listi, funkcija vraća nepromenjenu listu.

[Rešenje 3.35]

Primer 1

```
POKRETANJE: izbaci 2 [1,2,3,4,5]
IZLAZ:
 [1,2,4,5]
```

Primer 2

```
POKRETANJE: izbaci (-2) [1,2,3]
IZLAZ:
 [1,2,3]
```

Zadatak 3.36 Napisati funkciju `ubaci k n l` koja ubacuje u listu `l` na poziciju `k` element `n`. U slučaju da je zadata neispravna pozicija u listi, dodati element `n` na kraj liste.

[Rešenje 3.36]

Primer 1

```
POKRETANJE: ubaci 2 5 [1,2,3]
IZLAZ:
 [1,2,5,3]
```

Primer 2

```
POKRETANJE: ubaci 10 5 [1,2,3,4,5]
IZLAZ:
 [1,2,3,4,5,5]
```

3.2.3 Zadaci za vežbu

Zadatak 3.37 Ana uči prirodne brojeve i zanima je na koje sve načine može da razloži dati broj u obliku proizvoda dva prirodna broja, kao i koliko takvih razlaganja ima. Definisati sledeće funkcije koje pomažu Ani u rešavanju pomenutog problema:

a) `razlozi n` - za dati prirodan broj `n` vraća listu parova `(a,b)` takvih da su `a` i `b` prirodni brojevi iz intervala `[1,n]` i da je proizvod `a` i `b` jednak `n`

Primer 1

```
POKRETANJE: razlozi 3
IZLAZ:
 [(1,3),(3,1)]
```

Primer 2

```
POKRETANJE: razlozi 4
IZLAZ:
 [(1,4),(2,2),(4,1)]
```

b) `brojP n` - računa na koliko se različitih načina dati prirodan broj `n` može predstaviti u obliku proizvoda dva prirodna broja

Primer 1

```
POKRETANJE: brojP 3
IZLAZ:
 2
```

Primer 2

```
POKRETANJE: brojP 4
IZLAZ:
 3
```

Zadatak 3.38 Petar je u školi dobio zadatak da proveri da li u datoj listi parova postoji par oblika `(prethodnik, sledbenik)` za dati prirodan broj. Definisati sledeće funkcije koje pomažu Petru u rešavanju pomenutog problema:

a) `par n` - za dati prirodan broj `n` vraća par brojeva `(prethodnik,sledbenik)`

Primer 1

```
POKRETANJE: par 5
IZLAZ:
 (4,6)
```

Primer 2

```
POKRETANJE: par 12
IZLAZ:
 (11,13)
```

b) `postojiPar lista_parova` - za datu listu parova `(a,b)` i prirodan broj `n` proverava da li u listi postoji bar jedan par `(a,b)` takav da je `a` prethodnik broja `n`, a `b` sledbenik broja `n`

Primer 1

```
POKRETANJE: postojiPar [(1,4),(1,3)] 2
IZLAZ:
 True
```

Primer 2

```
POKRETANJE: postojiPar [(2,4),(2,3)] 2
IZLAZ:
 False
```

Zadatak 3.39 U toku je prijavljivanje za seminar na temu `IT problemi`. Zbog ograničenog broja mesta, sa liste prijavljenih se mogu primiti samo oni čiji je redni broj prijave manji od kapaciteta sale za održavanje seminara. Napisati funkciju `primljeni n lista` koja deli listu prijavljenih na dve liste na osnovu kapaciteta sale `n`, prva lista sadrži prvih `n` elemenata i predstavlja učesnike seminara, a druga lista sadrži ostatak i predstavlja prijavljenje koji se stavljaju na listu čekanja za sledeći termin održavanja.

Primer 1

```
POKRETANJE: primljeni 2 ["Marko Mitic", "Aleksa Jovic", "Andjela Antic"]
IZLAZ:
 [["Marko Mitic", "Aleksa Jovic"],["Andjela Antic"]]
```

Zadatak 3.40 Prodavnica `MiniMaxi` slavi 25 godina postojanja i tim povodom je organizovala posebnu akciju za svoje kupce. Svaki artikal vrednosti do 1000 dinara, čija je cena deljiva sa 25 se dobija u pola svoje cene. Napisati funkciju `usteda lista` koja na osnovu liste cena kupljenih artikala, računa koliku je uštedu ostvario kupac.

Primer 1

```
POKRETANJE: usteda [56.2,125,1025,658,300]
IZLAZ:
 212.5
```

Primer 2

```
POKRETANJE: usteda [56.2,125.99,102,658,326]
IZLAZ:
 0
```

Zadatak 3.41 Učesnici nagradne igre `Sedmica` zadaju kombinacije od sedam različitih pozitivnih brojeva manjih od 30, nakon čega se generiše dobitna kombinacija. Glavnu nagradu dobija učesnik koji je pogodio svih sedam brojeva, a utešne nagrade oni koji su imali veći broj pogodaka od promašaja u svojoj kombinaciji. Napisati funkcije:

a) `pogodak ucesnikL dobitnaL` - vraća par (a, b) takav da a predstavlja broj pogodaka, a b broj promašaja u kombinaciji učesnika koja je zadata listom `ucesnikL` na osnovu dobitne kombinacije, liste `dobitnaL`.

Primer 1

```
POKRETANJE: pogodak [2,12,15,17,19,25,29] [12,13,20,21,25,28,29]
IZLAZ:
 (3,4)
```

b) `nagrada ucesnikL dobitnaL` - vraća poruku o nagradi: `Sedmica`, `Utesna nagrada` ili `Vise srece drugi put`. Kombinacija učesnika je zadata listom `ucesnikL`, a dobitna kombinacija listom `dobitnaL`. Učesnik osvaja nagradu `Sedmica` ukoliko je pogodio dobitnu kombinaciju, utešnu nagradu ukoliko njegova kombinacija ima više pogodaka od promašaja, inače ne osvaja nista.

Primer 1

```
POKRETANJE: nagrada [9,13,15,17,19,21,23] [13,15,19,21,22,23,27]
IZLAZ:
 "Utesna nagrada"
```

3.3 Funkcije

3.3.1 Uvodni primeri

Zadatak 3.42 Napisati funkciju `spoji lista1 lista2` koja pravi listu uređenih parova tako što spaja redom elemente prve liste sa elementima druge liste u parove rezultujuće liste.

```
{-
 Funkcija zip pravi listu torki od dve liste
 tako sto spaja elemente prve liste sa elementima druge liste
-}

spoji lista1 lista2 = zip lista1 lista2
```

Zadatak 3.43 Napisati funkciju `uvecaj lista` koja svaki element celobrojne liste uvećava za jedan.

```
{-
  Funkcije viseg reda
  -funkcije koje uzimaju funkciju kao argument ili vracaju funkciju kao rezultat

  Funkcija map
  - kao prvi argument uzima funkciju koju primenjuje na sve elemente liste
  koju uzima kao drugi argument

  map :: (a -> b) -> [a] -> [b]

  Sekcije

  Operatore mozemo zapisati kao Karijeve funkcije:
```

```
  (+) x y <-> x+y
  (+2) x <-> x+2

  Generalno ako je op operator
  (op), (x op), (op y) nazivamo sekcijama

-}

uvecaj lista = map (+1) lista
```

Zadatak 3.44 Napisati funkciju `pozitivni lista` koja izdvaja sve pozitivne elemente iz liste.

```
{-
  Funkcija filter
  - kao prvi argument uzima funkciju uslova (funkcija koja vraca logicku vrednost)
  i izdvaja iz liste koju uzima kao drugi argument
  sve elemente koji zadovoljavaju zadat uslov

  filter :: (a -> Bool) -> [a] -> [a]

-}

pozitivni lista = filter (>0) lista
```

Zadatak 3.45 Napisati funkciju `prviNePozitivni lista` koja izdvaja najduži prefiks pozitivnih elemenata liste.

```
{-
  Jos neke funkcije viseg reda za rad sa listama:

  any uslov lista - vraca True ako postoji element u listi koji zadovoljava uslov,
    False inace
  all uslov lista - vraca True ako svi elementi u listi zadovoljavaju uslov, False
    inace
  takeWhile uslov lista - izdvaja jedan po jedan element liste sve dok ne stigne do
    elementa koji ne zadovoljava uslov
  dropWhile uslov lista - izbacuje jedan po jedan element liste sve dok ne stigne do
    elementa koji ne zadovoljava uslov
  sum lista - sabira elemente liste
  product lista - mnozi elemente liste
  reverse lista - obrce listu
  unzip - spaja listu parova u dve liste
  concat lista - spaja liste iz liste listi u jednu listu
  elem e lista - vraca True ako e postoji u listi, False inace
  replicate n x - kopira broj x n puta

-}

prviNePozitivni lista = takeWhile (<=0) lista
```

Zadatak 3.46 Napisati funkciju `sum lista` koja računa sumu elemenata celobrojne liste korišćenjem ugrađene funkcije `foldr` .

```
--  Bez koriscenja funkcije foldr

sum1 [] = 0 -- v = 0
sum1 (x:xs) = x + sum1 xs -- op = +

{-
  Funkcija foldr
  - enkapsulira sledeci sablon rekurzije

  f [] = v - f preslikava praznu listu u vrednost v
  f (x:xs) = x op (f xs) - nepraznu listu preslikava u funkciju op
  primenjenu na glavu liste (x) i f od repa liste (f xs)

  Npr:
-}

sum2 lista = foldr (+) 0 lista
```

Zadatak 3.47 Napisati funkciju `absSume lista_listi` koja na osnovu liste listi celih brojeva pravi listu apsolutnih suma elemenata liste listi korišćenjem kompozicije funkcija za rad sa listama.

```
{-
  Kompozicija funkcija

  (f . g) x  <-> f (g x)

-}

-- sum - ugradjena funkcija koja odredjuje sumu elemenata liste
-- abs - ugradjena funkcija za odredjivanje apsolutne vrednosti broja

absSume lista_listi = map (abs . sum) lista_listi
```

Zadatak 3.48 Napisati funkciju `sledbenici l` koja vraća listu sledbenika elemenata liste `l` koji su prirodni brojevi.

```
{-
  Cesto nam je potrebno da listu zadamo na sledeci nacin:

  lista = [f(x) | x<-D, p(x)]

  to mozemo jednostavno uraditi koristeci funkcije map i filter

  map f(filter p xs)
-}

-- sledbenik prirodnog broja = broj + 1

sledbenici [] = []
sledbenici = map (+1) (filter (>0) lista)
```

3.3.2 Zadaci za samostalni rad sa rešenjima

Zadatak 3.49 Ana uči brojanje i razlikovanje boja koristeći kutiju punu jednobojnih kuglica. Ona prvo žmureći iz kutije izvuče određeni broj kuglica i poređa ih u niz u redosledu izvlačenja. Zatim izabere proizvoljnu boju i odredi na kojoj se sve poziciji u nizu izvučenih kuglica nalazi kuglica željene boje. Napisati funkciju `pozicije x l` koja vraća listu pozicija elementa `x` u listi `l`.

[Rešenje 3.49]

Primer 1

```
Pokretanje: pozicije "bela" ["plava","bela","bela","ljubicasta","bela"]
Izlaz:
 [1,2,4]
```

Zadatak 3.50 Učesnici nagradne igre `Sedmica` mogu proveriti dobitak putem sajta, gde se dobitna kombinacija objavljuje odmah nakon izvlačenja. Brojevi iz dobitne kombinacije se, radi jednostavnije provere pogodaka, uvek prikazuju u rastućem redosledu. Napisati funkciju `qsort l` koja rastuće sortira listu `l` algoritmom `qsort`.

[Rešenje 3.50]

Primer 1

```
Pokretanje: qsort [4,5,2,11,29,38,9]
Izlaz:
 [2,4,5,9,11,29,38]
```

Primer 2

```
Pokretanje: qsort [7,1,10,15,30,32,20]
Izlaz:
 [1,7,10,15,20,30,32]
```

Zadatak 3.51 Milan obožava da sakuplja sličice fudbalera. Da bi jednostavnije pratio koje mu sličice iz kolekcije još uvek nedostaju, čuva ih rastuće sortirane po rednom broju. Planirao je da dopuni svoju kolekciju na narednoj razmeni sličica, pa za tu priliku želi da iz svoje kolekcije izbaci sve nepotrebne duplikate koje će poneti na razmenu. Napisati funkciju `brisiPonavljanja l` koja briše sva uzastopna ponavljanja elemenata u listi `l`.

[Rešenje 3.51]

Primer 1

```
POKRETANJE: brisiPonavljanja [4,5,5,2,11,11,11]
IZLAZ:
 [4,5,2,11]
```

Primer 2

```
POKRETANJE: brisiPonavljanja [10,10,10,11]
IZLAZ:
 [10,11]
```

Zadatak 3.52 Kasirka Mica mora da ručno kuca artikle na kasi jer se pokvario skener barkodova. Pomozite Mici da taj posao obavi što brže grupisanjem artikala iste vrste na pokretnoj traci. Napisati funkciju `podlistePonavljanja l` koja grupiše sva uzastopna ponavljanja nekog elementa liste `l` u podlistu tako da rezultat bude lista listi.

[Rešenje 3.52]

Primer 1

```
POKRETANJE: podlistePonavljanja ["jabuke","jogurt","jogurt","hleb"]
IZLAZ:
 [["jabuke"],["jogurt","jogurt"],["hleb"]
```

Zadatak 3.53 Napisati funkciju `broj lista` koja vraća broj određen ciframa koje se nalaze u listi čitajući ih sa početka ka kraju liste i funkciju `brojObrnut lista` koja vraća broj određen ciframa koje se nalaze u listi čitajući ih sa kraja ka početku liste.

[Rešenje 3.53]

Primer 1

```
POKRETANJE: broj [1,0,1]
IZLAZ:
 101
```

Primer 2

```
POKRETANJE: brojObrnut [0,5,2]
IZLAZ:
 250
```

Zadatak 3.54 U toku su prijave za plesno takmičenje parova. Pored nagrade za najbolji par, dodeljuju se i pojedinačne nagrade za najboljeg ženskog i muškog takmičara. Da bi žiri jednostavnije beležio poene i odredio nagrade, potrebno je da im se dostave, pored liste parova koji se takmiče, i liste samo muških, tj. samo ženskih takmičara, odvojeno. Napisati funkciju `listaUPar lista` koja pretvara listu parova u par dve liste, tako da prva lista sadrži prve elemente svih parova, a druga druge elemente svih parova pod pretpostavkom da je prvi u paru uvek ženska osoba, a drugi muška (implementacija funkcije unzip).

[Rešenje 3.54]

Primer 1

```
POKRETANJE: listaUPar [("Ivana","Milan"), ("Ana","Jovan"), ("Anica","Petar")]
IZLAZ:
 (["Ivana","Ana","Anica"], ["Milan","Jovan","Petar"])
```

Zadatak 3.55 Nakon još jedne košarkaške sezone, potrebno je sumirati rezultate svih igrača. Svaki igrač ima jedinstveni redni broj, pod kojim se u bazi, u listi prezimena, čuva njegovo prezime, a u listi pogodaka, ostvaren broj poena u sezoni. Uparivanjem odgovarajućih podataka, napraviti za svakog igrača jedinstveni par oblika `(prezime, poeni)`. Napisati funkciju `parOdListi lista1 lista2` koja pravi listu parova od dve liste, liste prezimena i liste pogodaka, tako da prvi element svakog para bude iz prve liste, a drugi element svakog para bude iz druge liste (implementacija funkcije zip).

[Rešenje 3.55]

Primer 1

```
POKRETANJE: parOdListi ["Mikic","Peric","Jovic"] [100,76,96]
IZLAZ:
 [("Mikic",100),("Peric",76),("Jovic",96)]
```

Zadatak 3.56 Formacija igre `kolo` je polukrug sa istaknutom ulogom prvog i poslednjeg igrača. Napisati funkciju `ucesljaj mIgraci zIgraci` koja pravi jednu formaciju za kolo naizmeničnim učešljavanjem igrača iz date grupe muških i ženskih igrača, `mIgraci` i `zIgraci`, redom.

[Rešenje 3.56]

Primer 1

```
POKRETANJE: ucesljaj ["Petar","Aleksa","Filip"] ["Milica","Jovana","Anica"]
IZLAZ:
 ["Petar","Milica","Aleksa","Jovana","Filip","Anica"]
```

3.3.3 Zadaci za vežbu

Zadatak 3.57 Aleksa i Luka žele da komuniciraju preko šifrovanih poruka koje predstavljaju listom niski. Aleksa šifruje željene podatke na sledeći način: ukoliko se niska koju šalje sastoji samo od cifara, na njen početak i kraj dodaje karakter `C`, ako se sastoji samo od malih slova, na njen početak i kraj dodaje karakter `S`, a inače na njen početak i kraj dodaje karakter `O`. Definisati sledeće funkcije koje pomažu Aleksi da pošalje Luki šifrovanu poruku:

a) `broj s` - za datu nisku `s` proverava da li su svi njeni karakteri cifre

Primer 1

```
POKRETANJE: broj "123"
IZLAZ:
 True
```

Primer 2

```
POKRETANJE: broj ['c','a','o']
IZLAZ:
 False
```

a) `mala s` - za datu nisku `s` proverava da li su svi njeni karakteri mala slova

Primer 1

```
POKRETANJE: mala "pozdrav"
IZLAZ:
 True
```

Primer 2

```
POKRETANJE: mala ['C','a','o']
IZLAZ:
 False
```

c) `sifruj ls` - datu listu niski `ls` transformiše na sledeći način: ukoliko se niska koju šalje sastoji samo od cifara, na njen početak i kraj dodaje karakter `C`, ako se sastoji samo od malih slova, na njen početak i kraj dodaje karakter `M`, a inače na njen početak i kraj dodaje karakter `O`.

Primer 1

```
POKRETANJE: sifruj ["11","maj","petak"]
IZLAZ:
 ["C11C","MmajM","MpetakM"]
```

Primer 2

```
POKRETANJE: sifruj ["poz","Poz","poZ"]
IZLAZ:
 ["MpozM","OPozO","OpoZO"]
```

Zadatak 3.58 Za razliku od Alekse i Luke, Ana i Milica imaju problem dešifrovanja podataka. Da bi Ana dešifrovala podatke koje joj Milica pošalje potrebno je da svaku nisku iz dobijene liste transformiše na sledeći način: ukoliko dobijena niska počinje cifrom, sa njenog početka izbaciti onoliko karaktera koliko ta niska ima cifara, a ukoliko dobijena niska počinje malim slovom, sa njenog početka izbaciti onoliko karaktera koliko ta niska ima malih slova. Pretpostaviti da će ovakvim dešifrovanjem Ana uvek dobiti ispravne izvorne podatke. Definisati sledeće funkcije koje pomažu Ani da dešifruje Miličine poruke:

a) `cifre s` - za datu nisku `s` vraća broj karaktera niske koji su cifre

Primer 1

```
POKRETANJE: cifre "11Maj"
IZLAZ:
 2
```

Primer 2

```
POKRETANJE: cifre ['m','a','j']
IZLAZ:
 0
```

b) `mala s` - za datu nisku `s` vraća broj karaktera niske koji su mala slova

Primer 1

```
Pokretanje: mala "petak"
Izlaz:
 5
```

Primer 2

```
Pokretanje: mala ['C','a','o']
Izlaz:
 2
```

c) `desifruj ls` - datu listu niski `ls` transformiše na sledeći način: ukoliko niska počinje cifrom, sa njenog početka izbacuje onoliko karaktera koliko ta niska ima cifara, a ukoliko dobijena niska počinje malim slovom, sa njenog početka izbacuje onoliko karaktera koliko ta niska ima malih slova.

Primer 1

```
Pokretanje: desifruj ["aaa11","1minamaj2017","101petak"]
Izlaz:
 ["11","maj2017","petak"]
```

Zadatak 3.59 Napisati funkciju `magicniParovi l` koja pravi listu parova čiji su prvi elementi, elementi liste prirodnih brojeva `l`, a drugi elementi odgovarajući magični brojevi elemenata liste `l`. Magičan broj prirodnog broja `n` se dobija kao proizvod njegovih cifara.

Primer 1

```
Pokretanje: magicniParovi [12,101,154]
Izlaz:
 [(12,2),(101,0),(154,20)]
```

Primer 2

```
Pokretanje: magicniParovi [1000,99,111,222]
Izlaz:
 [(1000,0),(99,81),(111,1),(222,8)]
```

Zadatak 3.60 Podaci o cenama artikala u prodavnici su zadati u obliku liste realnih brojeva. Definisati funkciju `prosek lista_cena` koja računa prosečnu cenu artikla u prodavnici.

Primer 1

```
Pokretanje: prosek [199.99, 125, 10.99, 45.99, 123.50]
Izlaz:
 101.09400000000001
```

3.4 Rešenja

Rešenje 3.8

```
-- Pretpostavljamo da funkciju koristimo samo za n>=1

proizvodPrvih n
      | n == 1 = 1
      | otherwise = n * proizvodPrvih (n-1)
```

Rešenje 3.9

```
-- Ispitujemo da li je broj prost tako sto pozovemo pomocnu funkciju
-- koja ispituje da li postoji neki broj pocev od 2 do n koji deli broj n
prost n = prost1 n 2

-- Proveravamo da li postoji broj koji deli n (pocev od broja 2 - poziv iz prethodne
    funkcije)
-- ukoliko je k == n to znaci da smo proverili za svaki broj od 2 do n-1
-- i ustanovili da nijedan od njih ne deli n tako da vracamo True kao indikator da
    broj n jeste prost
-- ukoliko je n deljivo sa k, vracamo False (n nije prost)
-- ukoliko n nije deljivo sa k, pozovemo funkciju rekurzivno za sledeci broj (k+1)
prost1 n k
  | n == k = True
  | n `mod` k == 0 = False
  | otherwise = prost1 n (k+1)
```

Rešenje 3.10

```
nzd a b
  | b == 0 = a
  | otherwise = nzd b (a `mod` b)
```

Rešenje 3.11

```
tipJednacine a b c
  | a == 0 = "Degenerisana"
  | (b*b - 4*a*c) == 0 = "Jedno resenje"
  | (b*b - 4*a*c) > 0 = "Dva resenja"
  | otherwise = "Nema resenja"
```

Rešenje 3.12

```
uDekadnu x osn = if x==0 then 0 else uDekadnu (x `div` 10) osn * osn + (mod x 10)

izDekadne x osn = if x==0 then 0 else izDekadne (x `div` osn) osn * 10 + (mod x osn)
```

Rešenje 3.13

```
-- Trazimo ceo deo korena broja x
-- trazimo prvi broj (pocev od 1) ciji je kvadrat veci od kvadrata naseg broja x
-- i kao rezultat vracamo broj koji je za jedan manji
ceoDeo x = ceoDeo1 x 1

ceoDeo1 x i
  | (i*i) > x = (i-1)
  | otherwise = ceoDeo1 x (i+1)
```

Rešenje 3.14

```
-- Pravimo listu prvih n elemenata harmonijskog reda (n>0)
-- ukoliko je n = 1, vracamo listu koja sadrzi prvi element harmonijskog reda
-- inace pozovemo funkciju rekurzivno za n-1
-- i na rezultat nadovezemo reciprocnu vrednost broja n
harm n
  | n == 1 = [1.0]
  | otherwise = harm (n-1) ++ [recip n]
```

Rešenje 3.15

```
-- Pravimo listu delioca broja n
-- ukoliko je n = 1, vracamo listu koja sadrzi samo 1
-- inace pozivamo pomocnu funkciju sa jos jednim parametrom
-- koji nam govori do kog potencijalnog delioca smo stigli
delioci n
    | n == 1 = [1]
    | otherwise = delioci1 n 2

-- Pravimo listu delioca broja n pocev od broja k
-- ukoliko smo stigli do broja n (k==n) vracamo praznu listu
-- inace proveravamo da li je k delioc broja n
-- ako jeste, stavljamo ga u listu i pozivamo funkciju rekurzivno
-- za k+1 (sledeci potencijalni delilac)
-- inace samo pozivamo funkciju rekurzivno za k+1
delioci1 n k
    | k == n = []
    | n `mod` k == 0 = [k] ++ (delioci1 n (k+1))
    | otherwise = delioci1 n (k+1)
```

Rešenje 3.16

```
-- Nadovezujemo listu2 na listu1 n puta
-- ukoliko je n == 1 na listu1 nadovezemo listu2 operatorom ++
-- inace pozovemo funkciju rekurzivno za n-1 i na rezultat nadovezemo listu2
nadovezi lista1 lista2 n
  | n == 1 = lista1 ++ lista2
  | otherwise = (nadovezi lista1 lista2 (n-1)) ++ lista2
```

Rešenje 3.29

```
bezbedanRep1 lista = if lista == [] then [] else tail lista

bezbedanRep2 lista
  | lista == [] = []
  | otherwise = tail lista

bezbedanRep3 [] = []
bezbedanRep3 (x:xs) = xs
-- bezbedanRep3 lista = tail lista
```

Rešenje 3.30

```
savrseni n = [x | x<-[1..n-1], sum(faktori x) == x]
faktori x = [i | i<-[1..x-1], x `mod` i ==0]
```

Rešenje 3.31

```
zbirPar n = [(a,b)| a<-[1..n], b<-[1..n], a+b==n]
-- drugi, efikasniji nacin
zbirPar2 n = [(a,b)| a<-[1..n], b<-[n-a], b/=0]
```

Rešenje 3.32

```
poslednji lista = lista !! poz
                where poz = length lista - 1  -- moze i: length(tail lista)

{-
poslednji [x] = x
poslednji (_:xs) = poslednji xs
-}
```

Rešenje 3.33

```
spoji [] = [] -- nije neophodno, prolazi drugi sablon i za praznu listu
spoji lista = [x | podlista <- lista, x <- podlista]

{-
spoji [] = []
spoji (x:xs) = x ++ spoji xs
-}
```

Rešenje 3.34

```
sufiksi [] = [[]]
-- lista je sama svoj sufiks, a ostali sufiksi su sufiksi njenog repa
sufiksi (x:xs) = (x:xs) : sufiksi xs
```

Rešenje 3.35

```
izbaci _ [] = []
izbaci 0 (x:xs) = xs
izbaci k (x:xs) = x : (izbaci (k-1) xs)
```

Rešenje 3.36

```
ubaci 0 n lista = n : lista
ubaci k n [] = [n]
ubaci k n (x:xs) = x : (ubaci (k-1) n xs)
```

Rešenje 3.49

```
-- Racunamo sve pozicije broja x u listi
-- tako sto spajamo listu sa listom brojeva od 0 do n
-- i od liste parova (broj, pozicija)
-- izdvajamo one pozicije kod kojih je broj = x
-- where omogucava da u okviru funkcije uvedemo novu promenljivu za neki izraz koji se koristi u definiciji
pozicije :: Eq a => a -> [a] -> [Int]
pozicije x [] = []
pozicije x lista = [i | (x1,i) <- zip lista [0..n], x == x1]
                  where n = length lista - 1
```

Rešenje 3.50

```
qsort :: Ord a => [a] -> [a]
qsort [] = []
qsort (x:xs) = qsort manji ++ [x] ++ qsort veci
            where manji = [a | a<-xs, a <= x]
                  veci = [b | b<-xs, b > x]
```

Rešenje 3.51

```
-- Pravimo listu bez uzastopnih ponavljanja
-- tako na glavu nadovezemo rezultat rekurzivnog poziva funkcije nad korigovanim repom
-- sa pocetka repa izbacujemo sve elemente koji su jednaki glavi
-- [1,1,1,1,1,2]
-- 1 : brojPonavljanja [2]
-- [1,2]
brisiPonavljanja :: Eq a => [a] -> [a]
brisiPonavljanja [] = []
brisiPonavljanja (x:xs) = x : brisiPonavljanja(dropWhile (==x) xs)
```

Rešenje 3.52

```
-- Pravimo listu koja sadrzi sva uzastopna pojavljivanja elemenata u posebnim listama
-- [1,1,1,2,2,3] -> [[1,1,1], [2,2], [3]]
-- tako sto pravimo listu uzastopnih pojavljivanja glave
-- i pozivamo funkciju rekurzivno pocev od elementa koji nije jednak glavi
-- [[1,1,1] , podlistePonavljanja [2,2,3]]
-- [[1,1,1], [2,2], podlistePonavljanja [3]]
-- [[1,1,1], [2,2], [3], podlistePonavljanja []]

podlistePonavljanja :: Eq a => [a] -> [[a]]
podlistePonavljanja [] = []
podlistePonavljanja (x:xs) = (x : (takeWhile (==x) xs)) : podlistePonavljanja(dropWhile (==x) xs)
```

Rešenje 3.53

```
-- [1,2,3] -> 321

brojObrnut :: Num a => [a] -> a
brojObrnut [] = 0
brojObrnut (x:xs) = (brojObrnut xs)*10 + x

-- [1,2,3] -> 123
-- Koriscenje funkcije brojObrnut
```

```
-- broj lista = brojObrnut (reverse lista)

-- Drugi nacin
broj [] = 0
broj (x:xs) = x*10^(length xs) + broj xs
```

Rešenje 3.54

```
-- Pravimo par listi od liste parova kao sto radi funkcija unzip
-- [(1,2), (1,2), (1,2)] -> ([1,1,1], [2,2,2])
-- tako sto svaki par iz liste dodajemo u akumulator cija je pocetna vrednost par
    praznih listi
-- [(1,2), (1,2), (1,2)]  - ([],[])
-- [(1,2), (1,2)]  - ([1],[2])
-- [(1,2)]  - ([1,1],[2,2])
-- []  - ([1,1,1],[2,2,2])
listaUPar :: [(a,b)] -> ([a],[b])
listaUPar [] = ([],[])
listaUPar lista = foldr (\ (a,b) (c,d) -> (a:c, b:d)) ([],[]) lista

{- Drugi nacin bez foldr:
listaUPar [] = ([],[])
listaUPar ((a,b):xs) = (a:l1, b:l2)
    where
      l1 = fst rep
      l2 = snd rep
      rep = listaUPar xs
-}
```

Rešenje 3.55

```
-- Implementacija funkcije zip
-- [1,2,3] [4,5,6] -> [(1,4), (2,5), (3,6)]
-- Da bi radila po duzini krace liste neophodno je dodati sablone za takve situacije
parOdListi [] _ = []
parOdListi _ [] = []
-- Od glava listi pravimo par koji dodajemo na pocetak rezultujuce liste za rep
parOdListi (x:xs) (y:ys) = ( (x, y) : (parOdListi xs ys))
```

Rešenje 3.56

```
-- [1,1,1] [2,2,2] -> [1,2,1,2,1,2]
-- [1,1] [2,2,2,2] -> [1,2,1,2,2,2]
-- Treba obuhvatiti slucajeve listi razlicitih duzina
-- Sledeca linija koda ne radi jer nije dozvoljeno koriscenje dzoker promenljive u
    izrazu kojim definisete vrednost funkcije
-- ucesljaj [] _ = _
ucesljaj [] lista = lista
ucesljaj lista [] = lista
ucesljaj (x:xs) (y:ys) = [x]++[y]++(ucesljaj xs ys)
```

4

Funkcionalni koncepti u programskom jeziku Python

Literatura:

(a) https://docs.python.org/2/howto/functional.html

4.1 Funkcionalno programiranje

4.1.1 Uvodni primeri

Zadatak 4.1 Napisati program koji sa standardnog ulaza učitava dva broja i na standardni izlaz ispisuje njihov zbir.

```
# jedan nacin definisanja funkcije suma
def suma(a, b):
    # ova funkcija je imperativna funkcija
    return a+b
# drugi nacin definisanja funkcije je putem anonimih funkcija (lambda izraza)
nova_suma = lambda a,b: a + b
a = int(raw_input("Unesite a:"))
b = int(raw_input("Unesite b:"))

print("Suma a i b je:")
print(nova_suma(a, b))
```

Zadatak 4.2 Napisati funkciju `parovi(a,b,c, d)` koja generiše listu parova celih brojeva (x, y), za koje `x` pripada segmentu $[a, b]$, a `y` pripada segmentu $[c, d]$. Testirati rad funkcije pozivom u programu.

```
# za pravljenje listi mozemo koristi generatore listi kao u Haskelu

a = int(raw_input("Unesite a:"))
b = int(raw_input("Unesite b:"))
c = int(raw_input("Unesite c:"))
d = int(raw_input("Unesite d:"))
print [(x,y) for x in range(a, b+1) for y in range(c, d+1)]
```

Zadatak 4.3 Marko, Petar i Pavle su polagali ispit iz predmeta programske paradigme. Napisati program koji sa standardnog ulaza učitava ocene koji su dobili, a potom ispisuje listu parova (student, ocena) na standardni izlaz.

```
# Funkcija zip pravi listu torki od dve liste
# tako sto spaja elemente prve liste sa elementima druge liste
lista1 = ["Marko", "Petar", "Pavle"]
lista2 = []
for osoba in lista1:
    ocena = int(raw_input("Unesite ocenu koju je ostvario " + osoba + " "))
    lista2.append(ocena)
```

```
print zip(lista1, lista2)
```

Zadatak 4.4 Napisati program koji sa standarnog ulaza učitava nisku, a na standardni izlaz ispisuje nisku u kojoj su sva mala slova pretvorena u velika.

```
# pseudo kod za map.
# def map(func, seq):
#     # vraca `Map` objekat gde
#     # funkcija func primenjena na svaki element
#     return Map(
#         func(x)
#         for x in seq
#     )

niska = raw_input("Ucitajte zeljenu nisku: ")
print ''.join(list(map(lambda slovo: slovo.upper(), niska)))
```

Zadatak 4.5 Napisati program koji sa standarnog ulaza učitava nisku, a na standardni izlaz ispisuje sve karaktere koji nisu slova.

```
# Funkcija filter
# - kao prvi argument uzima funkciju uslova (funkcija koja vraca logicku vrednost)
# i izdvaja iz liste koju uzima kao drugi argument
# sve elemente koji zadovoljavaju zadat uslov
# def filter(evaluate, seq):
#     return Map(
#         x for x in seq
#         if evaluate(x) is True
#     )

niska = raw_input("Ucitajte zeljenu nisku: ")
print list(filter(lambda slovo: slovo.isdigit(), niska))
```

Zadatak 4.6 Napisati program koji na standardni izlaz ispisuje sumu, koristeći funkciju `reduce` prvih n prirodnih brojeva gde se n unosi sa standardnog ulaza.

```
# funkcija koja odgovara funkciji fold u Haskelu je funkcija reduce
# ona se nalazi u modulu functools
from functools import reduce

n = int(raw_input("Unesite n: "))
brojevi = range(n+1)
print "Suma je: "
print reduce(lambda a, b: a+b, brojevi)
```

4.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 4.7 U datoteci `korpa.json` se nalazi spisak kupljenog voća u json formatu:

```
[ { 'ime' : ime_voca, 'kolicina' : broj_kilograma } , ...]
```

U datotekama `maxi_cene.json`, `idea_cene.json`, `shopngo_cene.json` se nalaze cene voća u json formatu:

```
[ { 'ime' : ime_voca, 'cena' : cena_po_kilogramu } , ...]
```

Napisati program koji izračunava ukupan račun korpe u svakoj prodavnici i ispisuje cene na standardni izlaz.

Primer 1

```
Pokretanje: python korpa.py
Interakcija sa programom:
 Maxi: 631.67 dinara
 Idea: 575.67 dinara
 Shopngo: 674.67 dinara
```

Sadržaj datoteka koje se koriste u primeru 4.7:

Listing 4.1: *korpa.json*

```
[ {"ime" : "jabuke" ,"kolicina": 3.3},
{"ime": "kruske" , "kolicina": 2.1},
{"ime": "grozdje" , "kolicina":  2.6},
```

Listing 4.2: *maksi_cene.json*

```
[ {"ime" :"jabuke", "cena" : 59.9},
{"ime" :"kruske", "cena" : 120},
{"ime" :"grozdje", "cena" : 70},
{"ime" :"narandze", "cena" : 49.9},
{"ime" :"breskve", "cena" : 89.9} ]
```

Listing 4.3: *idea_cene.json*

```
[ {"ime" :"jabuke", "cena" : 39.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"breskve", "cena" : 59.9} ]
```

Listing 4.4: *shopngo_cene.json*

```
[ {"ime" :"jabuke", "cena" : 69.9},
{"ime" :"kruske", "cena" : 100},
{"ime" :"grozdje", "cena" : 90},
{"ime" :"maline", "cena" : 290},
```

[Rešenje 4.7]

Zadatak 4.8 U datoteci *tacke.json* se nalaze podaci o tačkama u sledećem formatu.

Listing 4.5: *tacke.json*

```
[ {"teme":teme , "koordinate": [x, y]}, ...]
```

Napisati program koji učitava podatke o tačkama iz datoteke *tacke.json* i sortira i po udaljenosti od koordinatnog početka opadajuće. Program na standardni izlaz treba da ispiše tačku iz zadatog kvadranta, koja je najudaljenija od koordinatnog početka. Kvadrant se zadaje kao argument komandne linije.

Primer 1

```
POKRETANJE: python tacke.py 1
INTERAKCIJA SA PROGRAMOM:
 B
```

Sadržaj datoteke koja se koristi u primeru 4.8:

Listing 4.6: *tacke.json*

```
[ {"teme":"A" , "koordinate": [10.0, 1.1]},
{"teme":"B" , "koordinate": [1.0, 15.0]},
{"teme":"C" , "koordinate": [-1.0, 5.0]} ]
```

[Rešenje 4.8]

4.1.3 Zadaci za vežbu

Zadatak 4.9 U datoteci *knjige.json* se nalaze podaci o knjigama u knjižari u sledećem formatu.

Listing 4.7: *knjige.json*

```
[ {"oznaka":broj , "naziv_autor": naziv knjige i autor, "kolicina":
   kolicina, "cena": cena po komadu}, ...]
```

Napisati program koji ispisuje listu parova (oznaka, ukupna vrednost). Ukupna vrednost se sračunava na osnovu cene i broja primeraka, a povećava se za 10 dinara ukoliko ukupna cena knjiga sa istom oznakom ne prelazi minimalnu cenu od 100 dinara.

Primer 1

```
POKRETANJE: python knjige.py
INTERAKCIJA SA PROGRAMOM:
 [('34587', 163.8), ('98762', 284.0), ('77226', 108.85000000000001), ('88112', 84.97)]
```

Sadržaj datoteke koja se koristi u primeru 4.9:

Listing 4.8: *knjige.json*

```
[ {"oznaka": 34587, "naziv_autor": "Learning Python, Mark Lutz", "
   kolicina": 4, "cena": 40.95},
{"oznaka": 98762, "naziv_autor": "Programming Python, Mark Lutz", "
   kolicina": 5, "cena": 56.80},
{"oznaka": 77226, "naziv_autor": "Head First Python, Paul Barry", "
   kolicina": 3, "cena": 32.95},
{"oznaka": 88112, "naziv_autor": "Einfuhrung in Python3, Bernd Klein",
   "kolicina": 3, "cena": 24.99} ]
```

Zadatak 4.10 Datoteka `fudbaleri.json` sadrži podatke o fudbalerima (ime, nacionalnost i broj golova) u sledećem formatu:

```
[ { "Ime" : "Alexis Sanchez",  "Nacionalnost" :"Cile", "Golovi" : 17} ,
    ...]
```

(a) Definisati funkciju `uporedi` koristeći lambda izraz, koja poredi dva fudbalera po broju postignutih golova. Funkcija vraća -1, 0 ili 1 ukoliko je prvi fubaler postigao manji, jednak i veći broj golova u odnosu na drugog fudbalera

(b) Napisati program koji iz izabrane datoteke izdvaja fudbalere određene nacionalnosti i sortira ih rastuće po broju golova. Željena nacionalnost (npr. 'Engleska') i ime datoteke u kojoj se nalaze podaci o fudbalerima se zadaju kao argumenti komandne linije, a rezultat rada programa se upisuje u datoteku `izabrana_nacionalnost.json` (npr. Engleska_nacionalnost.json). U slučaju greške prilikom pokretanja programa, ispisati tekst `Greska` na standardni izlaz.

Primer 1

```
POKRETANJE: python 1.py Engleska arsenal.json
ENGLESKA_NACIONALNOST.JSON
 [{"Nacionalnost": "Engleska",
  "Ime": "Alex Oxlade-Chamberlain",
  "Golovi": 2},
  {"Nacionalnost": "Engleska",
  "Ime": "Theo Walcott", "Golovi": 8}]
```

Sadržaj datoteke koja se koristi u primeru 4.10:

Listing 4.9: *fudbaleri.json*

```

[{"Nacionalnost": "Cile", "Ime": "Alexis Sanchez", "Golovi" : 17 },
{"Nacionalnost": "Francuska", "Ime": "Olivier Giroud", "Golovi" : 8},
{"Nacionalnost": "Engleska", "Ime": "Theo Walcott", "Golovi" : 8},}
{"Nacionalnost": "Engleska", "Ime": "Alex Oxlade-Chamberlain", "Golovi"
    : 2},
```

```
6 {"Nacionalnost": "Francuska", "Ime": "Laurent Koscielny", "Golovi" : 2
    } ]
```

Zadatak 4.11 Datoteka `utakmice.json` sadrži podatke o utakmicama (timovi koji se takmiče i vreme početka utakmice) za svaki sport posebno u sledećem formatu:

```
1 [ { "Timovi": "Liverpool-Arsenal", "Vreme": "18:00"} , ...]
```

(a) Napisati funkciju `u_vremenskom_intervalu(utakmice, pocetak, kraj)` koja vraća listu utakmica čiji se početak nalazi u vremenskom opsegu početak kraj. Funkciju implementirati bez korišćenja petlji.

(b) Napisati program koji sa standardnog ulaza učitava podatke o početku i kraju vremenskog intervala u formatu *%H:%M* i iz datoteka učitava podatke o utakmicama za sve sportove, i na standardni izlaz ispisuje listu utakmica čije se vreme početka nalazi u opsegu unešenog vremenskog intervala.

Primer 1

```
Pokretanje: python 1.py
Ulaz:
  Unesite početak intervala: 17:00
  Unesite kraj intervala: 18:00
Izlaz:
  {"Timovi": "Liverpool-Arsenal", "Vreme": "18:00"}
  {"Timovi": "Milan-Cheivo", "Vreme": "17:00"}
  {"Timovi": "Koln-Bayern", "Vreme": "17:30"}
```

Sadržaj datoteke koja se koristi u primeru 4.11:

Listing 4.10: *utakmice.json*

```
1 [{"Timovi": "Liverpool-Arsenal", "Vreme": "18:00"},
2  {"Timovi": "Milan-Cheivo", "Vreme": "17:00"},
3  {"Timovi": "Eibar-Real Madrid", "Vreme": "18:30"},
4  {"Timovi": "Roma-Napoli", "Vreme": "16:15"},
5  {"Timovi": "Koln-Bayern", "Vreme": "17:30"} ]
```

Zadatak 4.12 Napisati program koji na standardni izlaz ispisuje apsolutne putanje do svih datoteka koje sadrže reč koja se unosi sa standardnog ulaza ili u slučaju da ni jedna datoteka ne sadrži zadatu reč ispisati poruku `Ni jedna datoteka ne sadrzi zadatu rec`. Progam napisati korišćenjem lambda izraza i bez korišćenja petlji

Primer 1

```
Pokretanje: python 1.py
Ulaz:
  Unesite zadatu rec: include
Izlaz:
  /home/student/programiranje2/ispit/1.c
  /home/student/programiranje2/ispit/2.c
  /home/student/programiranje2/ispit/3.c
```

Primer 1

```
Pokretanje: python 1.py
Ulaz:
  Unesite zadatu rec: Lenovo
Izlaz:
  Ni jedna datoteka ne sadrzi zadatu rec
```

4.2 Rešenja

Rešenje 4.7

```
import json
from functools import reduce

def cena_voca(prodavnica, ime_voca):
  for voce in prodavnica:
    if voce['ime'] == ime_voca:
      return voce['cena']

# Ucitavamo podatke iz datoteka
f = open('korpa.json', "r")
korpa = json.load(f)
f.close()

f = open('maxi_cene.json', "r")
maxi_cene = json.load(f)
f.close()

f = open('idea_cene.json', "r")
idea_cene = json.load(f)
f.close()

f = open('shopngo_cene.json', "r")
shopngo_cene = json.load(f)
f.close()

maxi_racun = reduce(lambda a, b: a + b,[ voce['kolicina']*cena_voca(maxi_cene, voce['
    ime']) for voce in korpa])
idea_racun = reduce(lambda a, b: a + b,[ voce['kolicina']*cena_voca(idea_cene, voce['
    ime']) for voce in korpa])
shopngo_racun = reduce(lambda a, b: a + b,[ voce['kolicina']*cena_voca(shopngo_cene,
    voce['ime']) for voce in korpa])

print "Maxi: " + str(maxi_racun) + " dinara"
print "Idea: " + str(idea_racun) + " dinara"
print "Shopngo: " + str(shopngo_racun) + " dinara"
```

Rešenje 4.8

```
import json
import math
import sys
from functools import partial

if len(sys.argv) == 1:
    print "Niste naveli dovoljno argumenta komandne linije"
    exit()

with open("tacke.json","r") as f:
    tacke = json.load(f)

# funkcija koja tacke x i y poredi po njihovoj udaljenosti od koordinatnog pocetka
def poredi(x,y):
    if (x[0]*x[0] + x[1]*x[1]) > (y[0]*y[0] + y[1]*y[1]):
        return 1
    else:
        return -1

# funkcija koja proverava da li tacka pripada kvadrantu

def pripada(kvadrant, tacka):
    koordinate = tacka["koordinate"]
    if kvadrant == 1:
        return koordinate[0] >=0 and koordinate[1] >=0
    elif kvadrant == 2:
        return koordinate[0] <=0 and koordinate[1] >=0
    elif kvadrant == 3:
        return koordinate[0] <=0 and koordinate[1] <=0
    elif kvadrant == 4:
        return koordinate[0] >=0 and koordinate[1] <=0
    else:
        print "Kvadrant nije ispravno unet"
```

```
        exit()

sortirane_tacke = sorted(tacke, poredi, lambda x: x["koordinate"], True)
kvadrant = int(sys.argv[1])
# sa partial fiksiramo jedan argument
# partial vraca objekat koji se moze pozvati kao funkcija

uslov = partial(pripada, kvadrant)
tacke_iz_kvadranta = filter(uslov, sortirane_tacke)

if len(tacke_iz_kvadranta) > 0:
    print tacke_iz_kvadranta[0]["teme"]
else:
    print "Trazena tacka ne postoji"
```

5

Konkurentno programiranje

5.1 Scala

Koristicemo jezik Scala-u (verzija 2.11) http://www.scala-lang.org/.

5.1.1 Potreban softver i alati

Potrebno instalirati:

- Intellij Idea
  (jetbrains.com/idea/)
- Java JDK8
  (oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
- Plugin `Scala` za `Intellij Idea` (instalacija iz okruženja)

5.1.2 Pravljenje projekta

Koristi se `sbt` (eng. Simple build tool) koji će da preuzima i konfiguriše Spark biblioteku za nas. Na slici 6.2 je prikazano kako u okruženju odabrati `sbt` projekat.
Za pravljenje `sbt` projekta **neophodno** je imati aktivnu internet konekciju.

5.1.3 Konfiguracija projekta

Odaberite gde želite da se napravi projekat. Preporučeno je da imate direktorijum u kojem čuvate sve `Intellij Idea` projekte (uglavnom je to `home/korisnik/IdeaProjects`).
Za `sbt` i jezik `Scala` odaberite:

- `sbt: 0.13.max (0.13.17)`
- `Scala: 2.10.max (2.10.7)`

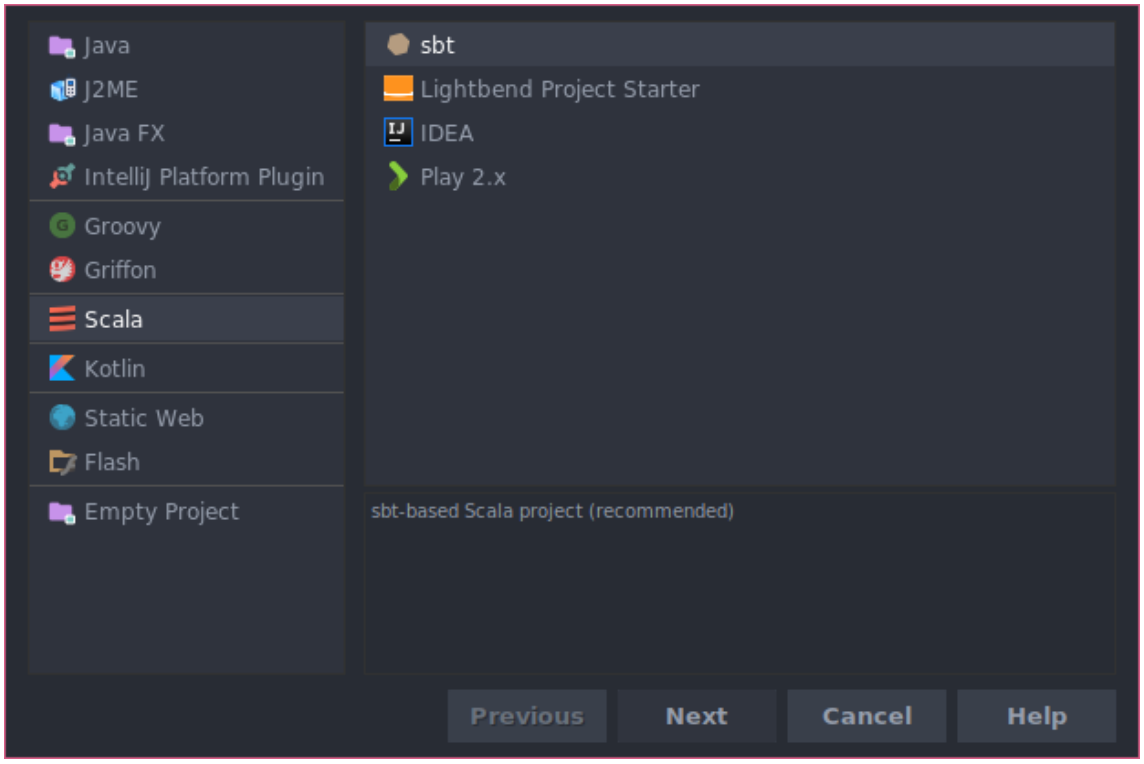
Na slici 6.3 prikazano je gde treba odabrati verzije za `sbt` i jezik `Scala`.
Kliknite `Finish`.

5.1.4 Inicijalizacija projekta

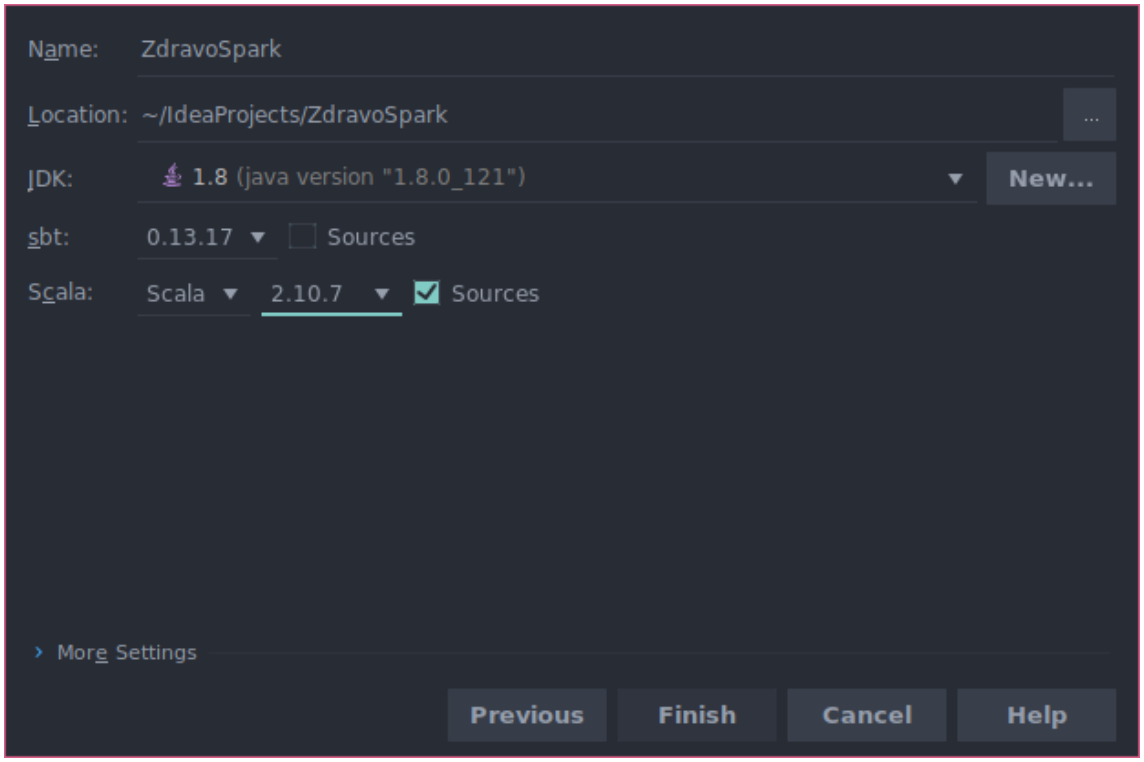
Okruženje se sada inicijalizuje, detektuju se Skala biblioteke i slično. Ovaj proces može da potraje od nekoliko sekundi do nekoliko minuta u zavisnosti od brzine mrežne konekcije i hardvera računara te treba biti strpljiv.
U donjem delu okruženja na sredini možete čitati poruke o tome šta se trenutno dešava, na primer:

```
sbt: dump project structure from sbt
```

je jedan od koraka koje Intellij mora da izvrši kako bi pripremio sve za rad.



Slika 5.1: *Početak pravljenja* `sbt` *projekta*



Slika 5.2: *Kraj pravljenja* `sbt` *projekta*

5.1.5 Uvod u jezik Scala

Zadatak 5.1 HelloWorld

```
/**
 * Viselinijske komentare u Scali pisemo ovako (isto kao i u C-u).
 *
 * */

// Jednolinijske komentare pisemo ovako

object HelloWorld {
  def main(args: Array[String]){
//  Tacka zapeta na kraju svake naredbe je opciona
    println("Hello world! :)")
  }
}
```

Zadatak 5.2 Promenljive, niske, petlje, funkcije

```

object Uvod {
    def main(args: Array[String]){

      println("------- Promenljive -------")
//    Promenljive mozemo deklarisati kao konstantne
//    ili promenljive cija se vrednost moze menjati.
//    Kljucna rec val deklarise konstantu a var promenljivu.
//
//    kljucnaRec imePromenljive : tipPromenljive = vrednost
//
      val c : Int = 42
      var x : Int = 6

      x += 6

//    Naredba:
//
//    c += 4
//
//    prijavljuje gresku.


      println("Konstanta: " + c)
      println("Promenljiva: " + x)

      if(x > 6)
        println("Promenljiva x je veca od 6.")
      else if(x == 6)
        println("Promenljiva x je jednaka 6.")
      else
        println("Promenljiva x je manja od 6.")

      println("------- Niske -------")

      val crtani : String = "Maza i Lunja"
      println("Duzina niske: '" + crtani + "' je " + crtani.length())

//    Viselinijske niske pisemo izmedju trostrukih navodnika:
      println(""" Likovi:
                  -Alisa
                  -Vendi
                  -Bambi
                  -Had
              """)
//    Interpolacija niski

//    Scala nam dozvoljava da ugnjezdavamo vrednosti promenljivih u niske
//    dodavanjem karaktera s na pocetak niske (npr: s"Moja niska") omogucavamo
//    ugnjezdavanje vrednosti promenljivih u nisku dodavajuci karakter $ pre imena
    promenljive
//    (npr s"Kolicina: $kol grama")
```

```
53        var trajanje : Int = 76
          println(s"Crtani film: '$crtani' - $trajanje minuta.")
55
          println("------- Petlje -------")
57
          var sat = 0
59
          while(trajanje >= 60){
61          sat += 1
            trajanje -= 60
63        }

65        println(s"'$crtani' traju $sat sat i $trajanje minuta.")

67  //    For petljom mozemo iterirati kroz kolekcije koristeci sintaksu
    //
69  //    for(element <- kolekcija)
    //
71  //    1 to 5 - pravi kolekciju brojeva [1,5]

73        println("FOR - 1 to 5 ")
          for(i <- 1 to 5)
75          println(i)

77  //    1 until 5 - pravi kolekciju brojeva [1,5)

79        println("FOR - 1 until 5 ")
          for(i <- 1 until 5)
81          println(i)

83  //    Range(pocetak, kraj, korak) - pravi kolekciju [pocetak, kraj) sa zadatik
        korakom.
    //    Korak moze biti i negativan npr. Range(10,0,-2)
85
          println("FOR - Range(0,6,2) ")
87        for(i <- Range(0,6,2))
            println(i)
89


91        println("------- Nizovi -------")
    //    Pravimo ih na sledeci nacin:
93  //      var niz : Array[tip] = new Array[tip](brojElemenata)
          var crtaci : Array[String] = new Array[String](5)
95        crtaci(0) = "Petar Pan"
          crtaci(1) = "Mulan"
97        crtaci(2) = "Aladdin"
          crtaci(3) = "Herkules"
99        crtaci(4) = "Pocahontas"

101       println("Crtaci: ")
          for(crtac <- crtaci)
103         println(crtac)

105       println("------- Funkcije -------")
          ispisiSortirano(crtaci)
107   }

109 //    def imeFunkcije([listaArgumenata]) : povratnaVrednost = {
    //          teloFunkcije
111 //    }

113 //  Povratna vrednost Unit je ekvivalentna void vrednosti
      def ispisiSortirano(crtaci : Array[String]) : Unit = {
115     println("Sortirani crtaci:")
        for(crt <- crtaci.sortWith(poredi))
117        println(crt)

119 //   Anonimne funkcije
    //
121 //    (listaArgumenata) => {teloFunkcije}

123       println("Sortirani crtaci koristeci anonimnu funkciju:")
```

```
      for(crt <- crtaci.sortWith((c1 , c2) => { if(c1.compareTo(c2) < 0) false else
    true} ))
125       println(crt)

127   }

129   def poredi(c1 : String, c2 : String) : Boolean = {
        if(c1.compareTo(c2) < 0)
131       return true
        else
133       return false
      }
135 }
```

Zadatak 5.3 Klase, objekti, nasledjivanje

```
2 //  Klase se u Scali konstruisu na sledeci nacin:
  //
4 //  class ImeKlase {
  //    teloKlase
6 //  }
  //      ili
8 //  class ImeKlase (argumentiKonstruktora) {
  //    teloKlase
10 //  }
  //
12
  class Film {
14   var naslov : String = ""
     var trajanje : Int = 0
16   var godina : Int = 0

18 // Konstruktor
     def this(nas : String, traj : Int, god : Int) = {
20     this()
       this.naslov = nas
22     this.trajanje = traj
       this.godina = god
24   }

26 // Metodi klase

28   def getNaslov() : String = {
       return this.naslov
30   }

32   def getTrajanje() : Int = {
       return this.trajanje
34   }

36   def getGodina() : Int = {
       return this.godina
38   }

40   override def toString() : String = {
       return "Film " + this.naslov + ", traje " + this.trajanje + " minuta, napravljen
       je " + this.godina + " godine"
42   }
  }
44
  // Nasledjivanje
46 class CrtaniFilm extends Film {
     var animator : String = ""
48
     def this(nas : String, traj : Int, god : Int, anim : String) = {
50     this()
       this.naslov = nas
52     this.trajanje = traj
       this.godina = god
54     this.animator = anim
     }
56
```

```
  def getAnimator() : String = {
58  return this.animator
  }
60
   override def toString() : String = {
62  return "Crtani " + super.toString() + ", animator je " + this.animator
  }
64 }

66 // U Scali mozemo definisati takozvane 'singleton' objekte kljucnom reci object.
// Garantuje se da ce postojati tacno jedan 'singleton' objekat na nivou naseg
    programa
68 // i on se najcesce koristi za implementaciju main metoda
object Program {
70
  def main(args: Array[String]) {
72
    val assassinsCreed = new Film("Assassin's Creed", 115, 2016)
74  val tarzan = new CrtaniFilm("Tarzan", 88 , 1999, "Walt Disney")

76  println(assassinsCreed)
    println(tarzan)
78
//  Scala nudi brojne korisne kontejnerske i nizovske klase
80 //  kao sto su Array, ArrayBuffer, Map, HashMap, Queue, Tuple1, Tuple2, i druge
//  sa kojima cemo se susretati u narednim primerima
82 //  gde ce biti vise objasnjene
  }
84
}
```

5.1.6 Zadaci

Zadatak 5.4 **Restoran** Pet konobara radi u restoranu, i nakon što dođu na posao, poslovođa im dodeli broj stolova koje moraju da usluže. Napisati program koji učitava sa standardnog ulaza broj neusluženih stolova u restoranu i raspoređuje ih konobarima. Nakon što usluže jedan sto, konobari šalju poruku tako da poslovođa u svakom trenutku ima uvid u brojčano stanje. Konobare implementirati kao posebne niti koje uslužuju sto (spavaju nasumično izabran broj sekundi), ispisuju redni broj stola koji su uslužili i po završetku ispisuju poruku da su završili.

[Rešenje 5.4]

Zadatak 5.5 **Nastavno osoblje** Koristeći biblioteku `HtmlCleaner` http://htmlcleaner.sourceforge.net/index.php napisati program koji dohvata imena, prezimena i email naloge nastavnog osoblja Matematičkog fakulteta sa sledecih stranica:

- http://www.matf.bg.ac.rs/redovni/
- http://www.matf.bg.ac.rs/vanredni/
- http://www.matf.bg.ac.rs/docenti/
- http://www.matf.bg.ac.rs/asistenti/
- http://www.matf.bg.ac.rs/saradnici/

i upisuje ih u odgovarajuće datoteke. Svaku stranicu obraditi u posebnoj niti.

[Rešenje 5.5]

Zadatak 5.6 **Množenje matrica** Napisati program koji konkurentno množi dve matrice čiji se elementi nalaze u datotekama `matrica1.txt` i `matrica2.txt` i rezultat upisuje u datoteku `matrica3.txt`. Svaka nit treba da računa vrednosti za jednu vrstu rezultujuće matrice. Format podataka u datotekama je sledeći:

```
n m
a11 a12 a13 ... a1m
a21 a22 a21 ... a2m
...
an1 an2 an3 ... anm
```

[Rešenje 5.6]

Zadatak 5.7 **Broj karaktera DNK** Napisati program koji konkurentno prebrojava koliko puta se koja baza (A, C, G, T) pojavljuje u DNK sekvenci koja se nalazi u višelinijskoj datoteci `bio_podaci.txt`. Sa standardnog ulaza učitati broj niti. Svakoj niti dodeliti određen broj linija koji ce da obrađuje a rezultate čuvati u deljenoj mapi (pogodno je koristiti klasu ConcurrentHashMap).

[Rešenje 5.7]

Zadatak 5.8 **Krediti** Bogoljub otvara banku i poseduje određeni kapital. Kao svaki dobar ekonomista on odlučuje da zaradi na davanju kredita. Napisati program koji demonstrira rad službenica i davanje kredita u Bogoljubovoj banci. Sa standardnog ulaza učitati njegov početni kapital, kamatnu stopu i broj zaposlenih službenica. U datoteci red_klijenata.txt se nalazi spisak klijenata koji čekaju na red za razgovor sa službenicom u sledećem formatu:

```
imeKlijenta potrebnaPozajmica
```

Posao svake službenice se izvršava u posebnoj niti. Službenica poziva sledećeg klijenta iz reda na razgovor. Nakon što ustanovi koliko je novca klijentu potrebno, službenica proverava da li banka trenutno poseduje tu količinu novca i daje klijentu kredit tako što umanjuje kapital banke i klijentu računa vrednost duga u skladu sa kamatom. Ukoliko banka nije u mogućnosti da izda kredit, ispisati odgovarajuću poruku. Nakon što službenice završe sa svim klijentima iz reda ispisati poruku o ukupnoj zaradi banke.

[Rešenje 5.8]

Zadatak 5.9 **Kladionica** Napisati program koji simulira proces klađenja. Kladioničari se klade na ishod pet fudbalskih utakmica uplaćujuči određenu količinu novca na tiket. U datoteci `kladionicari.txt` se nalazi spisak kladioničara i njihovih tiketa u sledećem formatu:

```
imeKladionicara svotaNovca
utakmica1
rezultat
 ...
utakmica5
rezultat
...
```

Ishod može biti 1 - prva ekipa je pobedila, x - nerešeno, 2 - druga ekipa je pobedila. Svaki ishod nosi određenu kvotu kojom se množi novac koji je kladioničar uplatio. U datoteci `utakmice.txt` se nalazi spisak utakmica sa njihovim kvotama u formatu:

```
 imeUtakmice
 kvota1 kvotaX kvota2
 ...
```

Učitati podatke o utakmicama i kladioničarima. Svaki kladioničar čeka na ishod utakmica u posebnoj niti. Kladionica nakon što dobije ishod utakmica (implementirati nasumično biranje ishoda) obaveštava kladioničare da su utakmice završene. Kladioničari nakon toga računaju sopstvenu zaradu kao zbir kvota pogođenih utakmica pomnoženih sa odgovarajućim delom novca i ispisuju poruku o pogodjenom rezultatu. Na kraju ispisati ukupnu količinu novca koju kladionica treba da isplati kladioničarima.

[Rešenje 5.9]

5.1.7 Zadaci za vežbu sa rešenjima

Zadatak 5.10 **Zbir vektora** Napisati program koji konkurentno sabira dva vektora. Sa standardnog ulaza se učitava dimenzija vektora, elementi oba vektora i broj niti. Svakoj niti dodeliti indeks početka i kraja vektora nad kojim računa zbir a rezultat smeštati u prvi vektor. Indekse računati na osnovu dimenzije vektora i broja niti. Rezultujući vektor ispisati na standardni izlaz.

[Rešenje 5.10]

Zadatak 5.11 **Broj petocifrenih brojeva** Napisati program koji konkurentno računa broj pojavljivanja petocifrenih brojeva koji se nalaze u datotekama čija imena se učitavaju sa standardnog ulaza. Svakoj niti dodeliti po jednu datoteku nad kojom će računati. Rezultate brojanja smeštati u lokalne promenljive unutar niti i po završetku računanja za svaku datoteku ispisati broj pojavljivanja petocifrenih brojeva.

[Rešenje 5.11]

Zadatak 5.12 **Berba** Milovan ima voćnjak u kome gaji trešnje, kajsije, kruške i šljive. Došlo je vreme berbe i mora uposliti mlade studente da oberu voćnjak. U datoteci `drvoredi.txt` se nalaze podaci o drvoredima voćnjaka u sledećem formatu:

```
1  vrstaVoca brojStabala
```

Sa standardnog ulaza učitati broj zaposlenih studenata a iz datoteke učitati podatke o drvoredima. Svaka nit predstavlja jednog studenta. Student odlazi do jednog drvoreda skidajući ga iz reda drvoreda koje je potrebno obrati i počinje branje. Ako pretpostavimo da jedno stablo voća može roditi od 30-50 kilograma voća, student za svako stablo iz drvoreda nasumično računa broj kilograma voća koji je obran i dodaje ih u skladište. Ukoliko su svi drvoredi obrani studenti prestaju sa radom i na standardni izlaz se ispisuje ukupna količina obranog voća svake vrste.

[Rešenje 5.12]

Zadatak 5.13 **Turistička agencija** Turistička agencija FlyProgrammer organizuje nagradnu igru i daje pet vaučera od 20% popusta na cenu kupljenje karte svojim klijentima. U datoteci `ucesnici.txt` se nalaze podaci o klijentima i cenama u sledećem formatu:

```
1  ime prezime
2  cena
```

Napisati program koji simulira nagradnu igru. Svaka nit čeka na rezultate nagradne igre za jednog klijenta. Turistička agencija izvlači dobitnike slučajnom selekcijom i nakon završenog izvlačenja obaveštava niti. Niti proveravaju da li je izvučen odgovarajući klijent i ispisuje poruku o ishodu.

[Rešenje 5.13]

5.1.8 Zadaci za vežbu

Zadatak 5.14 **Množenje vektora skalarom** Napisati program koji konkurentno množi vektor skalarom. Sa standardnog ulaza učitati dimenziju i elemente vektora, skalar i broj niti. Svakoj niti dodeliti deo vektora (indekse početnog i krajnjeg elementa). Nit računa proizvod vektora skalarom za elemente u zadatom opsegu i rezultat smešta u isti vektor. Na kraju ispisati rezultat na standardni izlaz.

Zadatak 5.15 **Množenje matrice vektorom** Napisati program koji konkurentno množi matricu vektorom. Sa standardnog ulaza učitati dimenziju i elemente vektora, dimenzije i elemente matrice i broj niti. Svakoj niti dodeliti deo matrice (indekse početne i krajnje vrste). Nit računa proizvod matrice vektorom za vrste u zadatom opsegu i rezultat smešta u nov vektor. Na kraju ispisati rezultat na standardni izlaz.

Zadatak 5.16 **Transponovanje matrice** Napisati program koji konkurentno transponuje matricu. U datoteci `matrica.txt` se nalaze dimenzije matrice nakon čega slede elementi matrice. Svaka nit transponuje po jednu vrstu matrice i smešta rezultat u rezultujuću matricu. Na kraju izračunavanja rezultujuću matricu upisati u datoteku `transponovana_matrica.txt`.

Zadatak 5.17 **Matrice (dodatak)** Implementirati zadatke 5.6 i 5.16 tako da se broj niti učitava sa standardnog ulaza, i svakoj niti dodeliti broj vrsta koje će obrađivati.

Zadatak 5.18 **Seminari konkursi projekti** Koristeći biblioteku `HtmlCleaner` napisati program koji konkurentno dohvata podatke o seminarima, konkursima i projektima Matematičkog fakulteta sa sledecih stranica:

- http://www.matf.bg.ac.rs/nauka-konkursi/
- http://www.matf.bg.ac.rs/seminari/
- http://www.matf.bg.ac.rs/projekti/

i upisuje ih u odgovarajuće datoteke. Svaku stranicu obraditi u posebnoj niti.

Zadatak 5.19 **Funkcije nad vektorima** Napisati program koji konkurentno računa proizvod, sumu, prosečnu vrednost, broj negativnih i broj pozitivnih elemenata vektora. Sa standardnog ulaza učitavati imena datoteka u kojima se nalaze vektori. Implementirati konkurentno računanje na dva načina, koristeći paralelizaciju zadataka i paralelizaciju podataka i uporediti vremena izvršavanja (obratiti pažnju na to da je korišćenjem paralelizacije zadataka najveće moguće ubrzanje jednako najsporijem zadatku dok kod paralelizacije podataka ubrzanje zavisi od broja procesora, broja niti,...). Na kraju ispisati rezultate svih izračunavanja na standardni izlaz.

Zadatak 5.20 **Broj petocifrenih brojeva (dodatak)** Implementirati zadatak 5.11 uz sledeće izmene. Brojeve učitati iz jedne datoteke `brojevi.txt` u niz a broj niti učitati sa standardnog ulaza. Svakoj niti dodeliti deo niza brojeva koji će obrađivati a rezultat čuvati u deljenoj promenljivoj tipa `AtomicLong`.

Zadatak 5.21 **Broj karaktera DNK (dodatak)** Implementirati zadatak 5.7 tako da se rezultati brojanja baza čuvaju u deljenim promenljivama tipa `AtomicLong` i na kraju ispisati procenat pojavljivanja svake baze u lancu.

Zadatak 5.22 **Hotel** Hotel Nightlife brine o svojim klijentima i trudi se da im smanji vreme čekanja dok se sređuje soba koju su rezervisali. U datoteci `sobe.txt` nalaze se brojevi soba koje je potrebno srediti. Sa standardnog ulaza učitati broj trenutno dostupnih čistačica. Svaka nit predstavlja jednu čistačicu. Čistačica skida sobe sa reda soba koje je potrebno srediti i odlazi do nje da je sredi (ispisati redni broj sobe u koju je ušla čistačica a nit uspavati na slucajan broj sekundi). Nakon što je sredila sobu, čistačica pronalazi ostavljen bakšiš (slučajan broj od 0-500 dinara) i po dogovoru bakšiš ubacuje u zajedničku kutiju. Kada završi sa jednom sobom, čistačica odlazi do sledeće sobe (skida je sa reda) i nastavlja sve dok red ne postane prazan. Nakon što završe sa čišćenjem, čistačice otvaraju kutiju sa bakšišem i dele novac na ravne časti tj. ispisuju na standardan izlaz dobijeni bakšiš.

Zadatak 5.23 **Loto** Napisati program koji simulira izvlačenje na Loto-u. Izvlače se tri broja iz opsege [1,37] a ukupna vrednost nagradnog fonda se unosi sa standardnog ulaza. Učesnici uplaćuju srećke i pokušavaju da pogode tri broja. U datoteci `ucesnici.txt` se nalaze informacije o uplaćenim srećkama u sledećem formatu:

```
1  ime
2  broj1 broj2 broj3
3  ...
```

Učitati podatke o učesnicima. Svaki učesnik čeka na kraj izvlačenja u posebnoj niti. Izvlačenje se odvija u glavnoj niti tako što se nasumično biraju tri broja iz opsega [1,37] i po završetku se obaveštavaju niti učesnika. Niti učesnika nakon toga porede izvučene brojeve i ukoliko se sva tri poklapaju ispisuju poruku i u ukupnu sumu novca koji je potrebno isplatiti dodaju vrednost nagradnog fonda. Ukoliko se dva od tri broja poklapaju u ukupnu sumu se dodaje 40% nagradnog

fonda. Ukoliko se samo jedan broj poklapa u ukupnu sumu se dodaje 10% nagradnog fonda. Na kraju ispisati količinu novca koju je potrebno isplatiti i količinu novca koju je lutrija zaradila (ukoliko zarada postoji).

5.2 Rešenja

Rešenje 5.4 Restoran

```
import java.util.concurrent._
import java.util.Scanner

// Pravljenje niti
//
// Da bismo napravili nit potrebno je da definisemo klasu koja nasledjuje klasu
    Thread
// i implementiramo metod run cije izvrsavanje pocinje kada nad instancom nase klase
// pozovemo metod start.
//
// Drugi nacin je da nasa klasa implementira interfejs Runnable
// i implementira metod run.
// Medjutim, da bismo naglasili da zelimo da se metod run nase instance izvrsava kao
    posebna nit
// potrebno je da napravimo instancu tipa Thread kojoj u konstruktoru treba da
    prosledimo instancu nase klase
// (koja implementira metod run.
//

class Konobar(ime : String, brStolova : Int) extends Thread {
  override def run(){
    for(i <- 0 until brStolova){
//    Klasa ThreadLocalRandom predstavlja generator slucajnih brojeva
//    jedinstven na nivou jedne niti.
//    Metod current() vraca objekat ove klase za trenutnu nit.
      Thread.sleep(ThreadLocalRandom.current().nextInt(1,10)*1000)
      println("Konobar " + ime + " je usluzio " + i + ". sto.")
    }
    println("Konobar " + ime + " je zavrsio sa posluzivanjem.")
  }
}

object Restoran {
   def main(args : Array[String]) {
     val sc : Scanner = new Scanner(System.in)

     println("Unesite broj neusluzenih stolova u restoranu: ")
     val brojStolova = sc.nextInt()

     val korak = Math.ceil(brojStolova/5.0).toInt
//   Pravimo instance niti
     val stefan = new Konobar("Stefan", korak)
     val nikola = new Konobar("Nikola", korak)
     val filip = new Konobar("Filip", korak)
     val nebojsa = new Konobar("Nebojsa", korak)
     val djordje = new Konobar("Djordje", brojStolova - 4*korak)

//   Ukoliko nasa klasa implementira interfejs Runnable
//
//   val stefan = new Thread(new Konobar("Stefan", 10))

//   Metod start zapocinje izvrsavanje metoda run
     stefan.start()
     nikola.start()
     filip.start()
     nebojsa.start()
     djordje.start()


   }
}
```

Rešenje 5.5 Nastavno osoblje

```
import java.net.URL
import java.util.concurrent._
import java.io.PrintWriter
import java.io.File

// Dodajemo biblioteku za parsiranje HTML stranica
// http://htmlcleaner.sourceforge.net/index.php
import org.htmlcleaner._

class ParserOsoblja(url : String, datoteka : String) extends Thread {

  override def run() {
//  Pravimo novi objekat tipa HtmlCleaner koji parsira HTML datoteku sa zadatim url-
    om
//  Neki od korisnih metoda klase HtmlCleaner su:
//    - clean(url) - vraca koreni cvor tipa TagNode sadrzaja datog url-a
//    - getInnerHtml(cvor) - vraca string - sadrzaj HTML koda datog cvora
//
//  Neki od korisnih metoda klase TagNode su:
//    - getChildTags() - vraca niz dece cvorova tipa TagNode
//    - hasChildren() - vraca true ukoliko cvor ima dece, false inace
//    - getAttributeByName(a)  - vraca vrednost atributa a
//    - getElementsByName(ime, rekurzivno)  - vraca niz dece sa zadatim imenom
//    - getElementsByAttValue(a, v, rekurzivno, caseSensitive)
//          - vraca niz dece koja imaju atribut a sa vrednoscu v
//    - getElementsHavingAttribute(a,rekurzivno)
//          - vraca niz dece koja imaju atribut a
//  ...
//
    val cleaner = new HtmlCleaner()
    val pw : PrintWriter = new PrintWriter(new File(datoteka))
//  Dohvatamo korenu etiketu - html
    val root = cleaner.clean(new URL(url))
//  Dohvatamo niz etiketa koje imaju klasu employer
    val elements = root.getElementsByAttValue("class", "employer", true, false)
    for(el <- elements){
//    Dohvatamo link- a etikete jer se u prvoj nalazi ime osobe
      val linkovi = el.getElementsHavingAttribute("href",true)
//    Dohvatamo div etikete jer se u prvoj nalazi korisnicko ime
      val divovi = el.getElementsByName("div", true)
      if(linkovi.length != 0 && divovi.length != 0)
        pw.append(linkovi(0).getText + " - " + divovi(0).getText + "@matf.bg.ac.rs \n
    ")
    }
    pw.close()
  }
}

object NastavnoOsoblje {
  def main(args : Array[String]) {
//  Pravimo niz niti koje ce da izvlace email adrese nastavnog osoblja na fakultetu
    val niti = new Array[ParserOsoblja](5)
    niti(0) = new ParserOsoblja("http://www.matf.bg.ac.rs/redovni/", "
    redovni_profesori.txt")
    niti(1) = new ParserOsoblja("http://www.matf.bg.ac.rs/vanredni/", "
    vanredni_profesori.txt")
    niti(2) = new ParserOsoblja("http://www.matf.bg.ac.rs/docenti/", "docenti.txt")
    niti(3) = new ParserOsoblja("http://www.matf.bg.ac.rs/asistenti/", "asistenti.txt
    ")
    niti(4) = new ParserOsoblja("http://www.matf.bg.ac.rs/saradnici/", "saradnici.txt
    ")

//  Pokrecemo niti
    for(i <- 0 until 5)
      niti(i).start()
  }
}
```

Rešenje 5.6 Množenje matrica

```
import java.util.concurrent._
import java.util.Scanner
import java.io.PrintWriter
import java.io.File
import scala.Array._

class Mnozilac(vrsta1 : Array[Int],
               matrica2 : Array[Array[Int]],
               rezultat : Array[Int])
               extends Thread {

  var k = matrica2.length*matrica2(1).length / vrsta1.length
  var m = vrsta1.length

  override def run() {
    for(i <- 0 until k)
      rezultat(i) = skProizvod(i)
  }

  def skProizvod(j : Int) : Int = {
    var res = 0
    for(i <- 0 until m)
      res += vrsta1(i)*matrica2(i)(j)
    return res
  }
}

object MnozenjeMatrica {
 def main(args : Array[String]) {

   val sc1 : Scanner = new Scanner(new File("matrica1.txt"))
   val sc2 : Scanner = new Scanner(new File("matrica2.txt"))
   val pw : PrintWriter = new PrintWriter(new File("matrica3.txt"))

// Ucitavamo dimenzije matrica
   val n = sc1.nextInt()
   val m1 = sc1.nextInt()
   val m2 = sc2.nextInt()
   val k = sc2.nextInt()

   if(m1 != m2){
     println("Greska! Dimenzije matrica se moraju poklapati!")
     return
   }

// Funkcija ofDim[Tip](n,m) pravi visedimenzioni niz dimenzija mxn
   val matrica1 = ofDim[Int](n,m1)
   val matrica2 = ofDim[Int](m2,k)
   val rezultat = ofDim[Int](n,k)

// Ucitavamo elemente prve matrice
   for(i <- 0 until n)
     for(j <- 0 until m1)
       matrica1(i)(j) = sc1.nextInt()

// Ucitavamo elemente druge matrice
   for(i <- 0 until m2)
     for(j <- 0 until k)
       matrica2(i)(j) = sc2.nextInt()

   val mnozioci = new Array[Mnozilac](n)

// Pravimo niz niti koje ce da racunaju i-tu vrstu rezultata mnozenja matrica
   for(i <- 0 until n)
       mnozioci(i) = new Mnozilac(matrica1(i), matrica2, rezultat(i))

// Zapocinjemo izvrsavanje niti
   for(i <- 0 until n)
     mnozioci(i).start()

// Cekamo da niti zavrse sa izracunavanjem
   for(i <- 0 until n)
     mnozioci(i).join()
```

```
75 // Upisujemo rezultujucu matricu u datoteku
   pw.append(s"$n $k \n")
77 for(i <- 0 until n){
     for(j <- 0 until k)
79     pw.append(s"${rezultat(i)(j)} ")
     pw.append("\n")
81 }

83 pw.close()
 }
85 }
```

Rešenje 5.7 Broj karaktera DNK

```
1  import java.util.concurrent._
   import java.util.concurrent.atomic._
3  import java.util.Scanner
   import java.io.File
5  import scala.collection.mutable.ArrayBuffer

7  class Brojac(poc : Int, kraj : Int,
               linije : ArrayBuffer[String],
9              mapaKaraktera : ConcurrentHashMap[Char, Int])
               extends Thread {
11
   override def run() {
13   for(i <- poc until kraj){
//     Racunamo broj svakog karaktera u liniji
15     val a = linije(i).count(_=='a')
       val c = linije(i).count(_=='c')
17     val g = linije(i).count(_=='g')
       val t = linije(i).count(_=='t')
19
//     Synchronized kljucna rec obelezava kriticnu sekciju
21 //   i garantuje se da u svakom trenutku tacno jedna nit moze izvsavati naredbe iz
       bloka.
   //   Synchronized se moze koristiti na vise nacina:
23 //
   //   Metodi klase
25 //
   //   def f() = synchronized { teloFunkcije }
27 //
   //   Na ovaj nacin smo naglasili da je metod f jedne instance nase klase kriticna
       sekcija
29 //   i u svakom trenutku tacno jedna nit moze izvrsavati ovaj metod te instance.
   //
31 //   Blok instance
   //
33 //   instanca.synchronized { blok }
   //
35 //   Na ovaj nacin smo naglasili da za datu instancu u svakom trenutku
   //   tacno jedna nit moze izvrsavati naredbe bloka
37 //   Ovakvo ponasanje mozemo posmatrati i iz drugacijeg ugla.
   //   Instanca predstavlja monitor objekat.
39 //   U trenutku kada nit pozeli da izvrsava blok kriticne sekcije,
   //   ona zakljuca instancu, izvrsi kriticnu sekciju i otkljuca instancu.
41 //
   //   Treba teziti ka tome da kriticna sekcija bude sto manja
43 //   kako bismo sto vise iskoristili prednosti konkurentnog izvrsavanja
   //
45 //   TODO: Zakomentarisati synchronized blok i videti sta se desava prilikom
       pokretanja programa
   //   sa razlicitim brojem niti.
47 //   Moguci scenario je da jedna nit procita vrednost iz mape, druga nit nakon toga
       azurira mapu,
   //   a prva nit i dalje drzi vrednost koju je procitala pre azuriranja tako da kada
       ona azurira mapu
49 //   rezultat azuriranja nece biti ispravan.
   //
51     mapaKaraktera.synchronized {
         mapaKaraktera.replace('a', mapaKaraktera.get('a')+a)
```

```
53          mapaKaraktera.replace('c', mapaKaraktera.get('c')+c)
            mapaKaraktera.replace('g', mapaKaraktera.get('g')+g)
55          mapaKaraktera.replace('t', mapaKaraktera.get('t')+t)
          }
57      }
    }
59  }

61  object BrojKarakteraDNK {
      def main(args : Array[String]) {
63      val sc1 : Scanner = new Scanner(new File("bio_podaci.txt"))
        val sc2 : Scanner = new Scanner(System.in)
65
        println("Unesite broj niti: ")
67      println("Broj procesora na racunaru koji su na raspolaganju je : " + Runtime.
        getRuntime().availableProcessors())

69      val brojNiti = sc2.nextInt()

71      val brojaci = new Array[Brojac](brojNiti)
//    Klasa ArrayBuffer predstavlja niz promenljive duzine
73  //  Neki od korisnih metoda su:
//      - append(e) - dodaje element na kraj niza
75  //    - isEmpty() - vraca true ukolik je niz prazan, false inace
//      - insert(i, e) - dodaje element na datu poziciju
77  //    ...
        val linije = new ArrayBuffer[String]()
79
        while(sc1.hasNextLine())
81        linije.append(sc1.nextLine())

83      val brojLinija = linije.length
        println(brojLinija)
85
//    Klasa ConcurrentHashMap predstavlja implementaciju mape cije su operacije
      bezbedne u kontekstu konkurentnog izvrsavanja.
87  //  To znaci da u svakom trenutku tacno jedna nit moze izvrsavati operacije nad mapom
      .
//    Medjutim, operacije citanja (get) su neblokirajuce, tako da se mogu preklopiti sa
       drugim operacijama (npr. azuriranja)
89  //  i u takvim slucajevima se ne garantuje azurnost rezultata.
//
91  //  Konstruktor prima tri argumenta:
//     - inicijalni kapacitet mape
93  //     - faktor povecavanja mape
//     - broj niti koji se pretpostavlja da ce konkurentno pristupati objektu mape
95  //
//    Neki od korisnih metoda klase ConcurrentHashMap su:
97  //     - get(kljuc) - vraca element sa zadatim kljucem, odnosno null ukoliko takav ne
      postoji
//     - put(kljuc, vrednost) - dodaje element sa zadatim parametrima
99  //     - remove(kljuc) - uklanja element sa zadatim kljucem
//     - replace(kljuc, vrednost) - postavlja vrednost elementu sa zadatim kljucem
101 //     - size() - vraca velicinu mape
//     - isEmpty() - vraca true ukoliko je mapa prazna, false inace
103 //     ...
//
105     val mapaKaraktera = new ConcurrentHashMap[Char, Int](4,4,brojNiti)
        mapaKaraktera.put('a', 0)
107     mapaKaraktera.put('c', 0)
        mapaKaraktera.put('g', 0)
109     mapaKaraktera.put('t', 0)

111     val korak = Math.ceil(brojLinija.toDouble/brojNiti.toDouble).toInt

113     for(i <- 0 until brojNiti)
          brojaci(i) = new Brojac(i*korak, Math.min((i+1)*korak, brojLinija), linije,
        mapaKaraktera)
115
        for(b <- brojaci)
117       b.start()

119     for(b <- brojaci)
```

```
      b.join()

    println("Rezultati konkurentnog izvrsavanja")
    println("A: " + mapaKaraktera.get('a'))
    println("C: " + mapaKaraktera.get('c'))
    println("G: " + mapaKaraktera.get('g'))
    println("T: " + mapaKaraktera.get('t'))

    println("Pravi rezultati \nA: 1761 \nC: 1577 \nG: 1589 \nT: 1913")
  }
}
```

Rešenje 5.8 Krediti

```
import java.util.concurrent.atomic._
import java.util.concurrent._
import java.util.Scanner
import java.io.File

class Sluzbenica(kamata : Int,
                 kapital : AtomicLong,
                 redKlijenata : ConcurrentLinkedQueue[Klijent],
                 zaduzeniKlijenti : ConcurrentLinkedQueue[Klijent])
                  extends Thread {

  override def run() {
     while(true){
//      Dohvatamo sledeceg klijenta iz reda
        var k : Klijent = redKlijenata.poll()
//      Ukoliko takav ne postoji zavrsavamo
        if(k == null)
          return

        println("Klijent " + k.getIme() + " razgovara sa sluzbenicom.")
        Thread.sleep(ThreadLocalRandom.current().nextInt(1,10)*1000)
//      Iako je kapital objekat AtomicLong i garantuje se atomicnost operacija azuriranja
//      mogu nastati problemi prilikom konkurentnog pristupanja i azuriranja,
//      i zbog toga je potrebno operacije sa ovim objektom obmotati synchronized blokom.
//      Problem moze nastati u delu koda (******).
//      Pretpostavimo da dve niti izvrsavaju ovaj deo koda konkurentno.
//      Prva nit procita vrednost kapitala, nakon toga druga nit procita vrednost kapitala
//      pre nego sto je prva nit izmenila vrednot, odmah posle prva nit promeni vrednost kapitala
//      i postavi novu vrednot (staraVrednost - prvaPozajmica)/
//      U ovom trenutku druga nit ima vrednost kapitala koja nije ispravna sa kojom dalje operise.
//      Druga nit promeni vrednost kapitala i postavi novu vrednost (staraVrednost - drugaPozajmica)
//      sto nije realna vrednost (staraVrednost - prvaPozajmica - drugaPozajmica)
        kapital.synchronized {
          if(k.getPozajmica() > kapital.get())
            println("Klijent " + k.getIme() + " ne moze dobiti kredit.")
          else{
            k.setDug(k.getPozajmica()*((100+kamata.toFloat)/100))
//          ******
            val novKapital = kapital.get() - k.getPozajmica()
            kapital.set(novKapital)
//          ******
            println("Klijent " + k.getIme() + " je dobio kredit u iznosu od "
                    + k.getPozajmica() + "e odnosno sa kamatom " + k.getDug() + "e.")
            zaduzeniKlijenti.add(k)
          }
        }
     }
  }
}

class Klijent(ime : String, pozajmica : Int) {
```

```
53    var dug : Float = 0

55    def getIme() : String = {
        return ime
57    }

59    def getPozajmica() : Int = {
        return pozajmica
61    }

63    def getDug() : Float = {
        return dug
65    }

67    def setDug(d : Float) = {
        dug = d
69    }
  }
71
  object Banka {
73    def main(args : Array[String]) {
  //    Klasa AtomicLong predstavlja enkapsulaciju long integer vrednosti
75  //    nad kojom se operacije azuriranja izvrsavaju atomicno.
  //
77  //    Neki od korisnih metoda ove klase su:
  //     - get() - vraca trenutnu vrednost
79  //     - set(v) - postavlja vrednost v
  //     - getAndAdd(v) - atomicki dodaje vrednost v i vraca prethodnu vrednost
81  //     - addAndGet(v) - atomicki dodaje vrednost v i vraca novu vrednost
  //     - getAndIncrement() - atomicki inkrementira vrednost i vraca prethodnu vrednost
83  //     - incrementAndGet() - atomicki inkrementira i vraca novu vrednost
  //     - getAndDecrement() - atomicki dekrementira vrednost i vraca prethodnu vrednost
85  //     - decrementAndGet() - atomicki dekrementira i vraca novu vrednost
  //     - compareAndSet(ocekivanaVrednost, novaVrednost) - postavlja novu vrednost
87  //       ukoliko je stara jednaka ocekivanoj
  //
89  //     U paketu java.util.concurrent.atomic postoje i druge korisne klase kao sto su
  //     AtomicBoolean, AtomicIntegerArray, AtomicInteger itd.
91      val sc1 : Scanner = new Scanner(System.in)

93      println("Unesite pocetni kapital banke: ")
        val kapital = new AtomicLong(sc1.nextLong())
95      val sacuvanKapital : Float = kapital.get()

97      println("Unesite kamatnu stopu: ")
        val kamata = sc1.nextInt()
99
        println("Unesite broj sluzbenica u ekspozituri: ")
101     val sluzbenice = new Array[Sluzbenica](sc1.nextInt())

103     val sc2 : Scanner = new Scanner(new File("red_klijenata.txt"))

105 //     Klasa ConcurrentLinkedQueue predstavlja implementaciju reda
    //      cije su operacije bezbedne u kontekstu konkurentnog izvrsavanja.
107 //
    //     Neki od korisnih metoda su:
109 //      - add(e) - dodaje element u red
    //      - poll() - skida element sa pocetka reda i vraca ga kao rezultat
111 //      - peek() - vraca elelent sa pocetka reda (ne skida ga)
    //      - remove(e) - ukljanja element e iz reda
113 //      - isEmpty() - vraca true ukoliko je red prazan
    //      ...
115 //
        val redKlijenata = new ConcurrentLinkedQueue[Klijent]()
117     val zaduzeniKlijenti  = new ConcurrentLinkedQueue[Klijent]()

119     while(sc2.hasNextLine())
          redKlijenata.add(new Klijent(sc2.next(), sc2.nextInt()))
121
        for(i <- 0 until sluzbenice.length)
123       sluzbenice(i) = new Sluzbenica(kamata, kapital, redKlijenata, zaduzeniKlijenti)

125     for(s <- sluzbenice)
```

```
      s.start()

    for(s <- sluzbenice)
      s.join()

//  Iteriramo kroz red zaduzenih klijenata i racunamo ukupno zaduzenje
    var ukupnoZaduzenje : Float = 0
    val iterator = zaduzeniKlijenti.iterator()
    while(iterator.hasNext())
      ukupnoZaduzenje += iterator.next().getDug()

    println(s"Banka je zaradila ${ukupnoZaduzenje-sacuvanKapital}e.")
  }
}
```

Rešenje 5.9 Kladionica

```
import java.util.concurrent.atomic._
import java.util.concurrent._
import java.util.Scanner
import java.io.File
import scala.collection.mutable.HashMap
import scala.collection.mutable.ArrayBuffer

class Kladionicar(ime : String,
                  novac : Int,
                  tiket : HashMap[String, Char],
                  utakmice : HashMap[String, Tuple4[Float,Float,Float,Char]])
                  extends Thread {

  var zarada : Float = 0

  override def run(){
//  Cekamo da se odigraju sve utakmice
//  Funkcije wait(), notify() i notifyAll()
//  moraju biti zakljucane unutar bloka synchronized
    utakmice.synchronized {
      utakmice.wait()
    }

    var pogodjeno = 0
    var ukupnaKvota : Float = 0
//  Racunamo ukupnu zaradu
    for(t <- tiket)
      if(t._2 == utakmice(t._1)._4){
        println(ime + " je pogodio utakmicu " + t._1 + " - " + utakmice(t._1)._4)
        pogodjeno += 1
        if(utakmice(t._1)._4 == '1')
          ukupnaKvota += utakmice(t._1)._1
        else if(utakmice(t._1)._4 == 'x')
          ukupnaKvota += utakmice(t._1)._2
        else if(utakmice(t._1)._4 == '2')
          ukupnaKvota += utakmice(t._1)._3
      }
     if(pogodjeno != 0)
       zarada = ukupnaKvota * novac/pogodjeno
  }

  def getIme() : String = {
    return ime
  }

  def getZarada() : Float = {
    return zarada
  }
}

object Kladionica {
  def main(args : Array[String]) {
    val sc1 : Scanner = new Scanner(new File("utakmice.txt"))
    val sc2 : Scanner = new Scanner(new File("kladionicari.txt"))
//  Klasa HashMap iz paketa scala.collection.mutable
```

```
//  predstavlja implementaciju mape koja se moze azurirati (eng. mutable)
//
//  Neke od korisnih funkcija su:
//    -put(k,v) - dodaje vrednost u mapu sa zadatim kljucem
//    -size - vraca velicinu mape
//    -contains(k) - vraca true ukoliko postoji element sa zadatim kljucem, false inace
//  ...
//
//  Takodje mozemo iterirati kroz elemente mape for petljom
//
//  Klasa Tuple4 je jedna u nizu klasa koje implementiraju torke (Tuple1, Tuple2, Tuple3,...)
//  koje se mogu azurirati.
//  Elementima torke pristupamo na sledeci nacin - torka._1, torka._2, torka._3, torka._4
//
    val utakmice = new HashMap[String, Tuple4[Float,Float,Float,Char]]()

//  Rezultate utakmica postavljamo da budu karakter '-'
//  kako bismo naglasili da se utakmice jos nisu odigrale
    while(sc1.hasNextLine()){
      utakmice.put(sc1.nextLine(),(sc1.nextFloat(), sc1.nextFloat(), sc1.nextFloat(), '-'))
      sc1.nextLine()
    }
    val kladionicari = new ArrayBuffer[Kladionicar]()

    while(sc2.hasNextLine()){
      val ime = sc2.next()
      val novac = sc2.nextInt()
      val tiket = new HashMap[String, Char]()
      for(i <- 0 until 5){
        sc2.nextLine()
        tiket.put(sc2.nextLine(), sc2.next()(0))
      }
      kladionicari.append(new Kladionicar(ime, novac, tiket, utakmice))
    }

    for(k <- kladionicari)
      k.start()

    println("Cekamo da se utakmice odigraju.")
    Thread.sleep(5000)

//  Racunamo rezultate utakmica
      val res = Array('1','x', '2')
      for(u <- utakmice)
        utakmice(u._1) = (u._2._1,
                          u._2._2,
                          u._2._3,
                          res(ThreadLocalRandom.current().nextInt(0, 3))
                          )

//  Ulazimo u kriticnu sekciju
//  i obavestavamo niti koje cekaju
    utakmice.synchronized {
      utakmice.notifyAll()
    }

    for(k <- kladionicari)
      k.join()

    var isplata : Float = 0
    for(k <- kladionicari){
      isplata += k.getZarada()
      println(k.getIme() + " ceka na isplatu " + k.getZarada() + "dinara.")
    }
    println("Ukupno kladionica treba da isplati " + isplata + " dinara.")
  }
}
```

Rešenje 5.10 Zbir vektora

```
import java.util.Scanner

class Sabirac(poc : Int,
              kraj : Int ,
              vektor1 : Array[Float],
              vektor2 : Array[Float])
              extends Thread {
// Svaka nit racuna zbir svog dela vektora [poc, kraj)
// i rezultat smesta u prvi vektor.
  override def run() {
    for( i <- poc until kraj)
      vektor1(i) += vektor2(i)
  }
}

object ZbirVektora {
  def main(args : Array[String]){

    val sc = new Scanner(System.in)

    println("Unesite dimenziju vektora: ")
    val n = sc.nextInt()

    val vektor1 : Array[Float] = new Array(n)
    val vektor2 : Array[Float] = new Array(n)

    println("Unesite elemente prvog vektora: ")
    for(i <- 0 until n)
        vektor1(i) = sc.nextFloat()

    println("Unesite elemente drugog vektora: ")
    for(i <- 0 until n)
        vektor2(i) = sc.nextFloat()

    println("Unesite broj niti: ")
    val brojNiti = sc.nextInt()

    val niti = new Array[Sabirac](brojNiti)

    val korak = Math.ceil(n/brojNiti.toDouble).toInt

//  Pravimo niti i zadajemo im indekse - granice
    for(i <- 0 until brojNiti)
        niti(i) = new Sabirac(i*korak, Math.min((i+1)*korak,n),vektor1,vektor2)

//  Pokrecemo racunanje
    for(i <- 0 until brojNiti)
        niti(i).start()

//  Cekamo da niti zavrse sa racunanjem
    for(i <- 0 until brojNiti)
        niti(i).join()

//  Kada sve niti zavrse sa racunanjem ispisujemo rezultat
    print("Zbir vektora je: \n[")
    for(i <- 0 until n-1)
      print(vektor1(i) + ", ")
    println(vektor1(n-1) + "]")
  }
}
```

Rešenje 5.11 Broj petocifrenih brojeva

```
import java.util.concurrent._
import java.io._
import java.util.Scanner
import scala.collection.mutable.ArrayBuffer

class BrojacPetocifrenih(dat : String) extends Thread {
```

```
  var rezultat : Int = 0
// Citamo brojeve iz datoteke i povecavamo lokalni brojac
// ukoliko je broj petocifren
  override def run() {
    val sc : Scanner = new Scanner(new File(dat))

    while(sc.hasNextInt()){
      var broj = sc.nextInt()
      if(broj >= 10000 && broj <= 99999)
        this.rezultat+=1
    }
  }

  def getRezultat() : Int = {
    return rezultat
  }

  def getDatoteka() : String = {
    return dat
  }
}

object BrojPetocifrenih {
  def main(args : Array[String]){

     val sc = new Scanner(System.in)

     val brojaci = ArrayBuffer[BrojacPetocifrenih]()
     var kraj = false
     var odg = ""
     var dat = ""

     while(!kraj) {
       println("Da li zelite da zadate ime datoteke koja ce biti obradjena (y/n)?")
       odg = sc.next()
       if(odg.toLowerCase() == "n")
         kraj = true
       else{
         println("Unesite ime datoteke: ")
         dat = sc.next()
         brojaci.append(new BrojacPetocifrenih(dat))
       }
     }

//   Zapocinjemo izvrsavanje
     for(brojac <- brojaci)
       brojac.start()

//   Pozivom metoda join cekamo sve brojace da zavrse sa izracunavanjem
     for(brojac <- brojaci)
       brojac.join()

//   Citamo rezultate brojanja svake niti
     for(brojac <- brojaci)
       println(s"Datoteka ${brojac.getDatoteka()} sadrzi ${brojac.getRezultat()}
    petocifrenih brojeva.")
  }
}
```

Rešenje 5.12 Berba

```
import java.util.concurrent.atomic._
import java.util.concurrent._
import java.util.Scanner
import java.io.File

class Berac(drvoredi : ConcurrentLinkedQueue[Tuple2[String,Int]],
            skladiste : AtomicIntegerArray) extends Thread {

  override def run() {

    while(true) {
```

```
//    Dohvatamo dvored za berbu iz reda
      val drvored = drvoredi.poll()
//    Ukoliko nema vise drvoreda za berbu zavrsavamo
      if(drvored == null)
        return
      println("Berac bere drvo " + drvored._1 )
      Thread.sleep(ThreadLocalRandom.current().nextInt(1,10)*1000)
//    Dodajemo kilograme voca koje smo obrali u skladiste
      for(i <- 0 until drvored._2){
        val obrano = ThreadLocalRandom.current().nextInt(30, 50)
        if(drvored._1 == "tresnje")
          skladiste.getAndAdd(0, obrano)
        else if(drvored._1 == "kruske")
          skladiste.getAndAdd(1, obrano)
        else if(drvored._1 == "kajsije")
          skladiste.getAndAdd(2, obrano)
        else if(drvored._1 == "sljive")
          skladiste.getAndAdd(3, obrano)
      }
    }
  }
}

object Berba {
  def main(args : Array[String]) {
    val sc1 : Scanner = new Scanner(new File("drvoredi.txt"))
    val sc2 : Scanner = new Scanner(System.in)

//  Pravimo skladiste voca koje imamo u vocnjaku
//  i postavljamo inicijalne kolicine voca.
//  Klasa AtomicIntegerArray sadzi niz integer vrednosti
//  nad kojima se operacije izvrsavaju atomicno.
//  Slicno kao kod klasa AtomicInteger, AtomicLong i dr.
    val skladiste = new AtomicIntegerArray(4)
    skladiste.set(0,0)
    skladiste.set(1,0)
    skladiste.set(2,0)
    skladiste.set(3,0)
//  Pravimo red drvoreda za berbu
//  Svaki drvored je jedan par (voce, brojStabala).
    val drvoredi = new ConcurrentLinkedQueue[Tuple2[String, Int]]()
    while(sc1.hasNextLine())
      drvoredi.add((sc1.next(), sc1.nextInt()))

    println("Unesite broj beraca: ")
    val brojBeraca = sc2.nextInt()

    val beraci = new Array[Berac](brojBeraca)
    for(i <- 0 until brojBeraca)
      beraci(i) = new Berac(drvoredi, skladiste)

    for(b <- beraci)
      b.start()

    for(b <- beraci)
      b.join()

    println("Tresanja je obrano: " + skladiste.get(0) + " kilograma.")
    println("Krusaka je obrano: " + skladiste.get(1) + " kilograma.")
    println("Kajsija je obrano: " + skladiste.get(2) + " kilograma.")
    println("Sljiva je obrano: " + skladiste.get(3) + " kilograma.")
  }
 }
```

Rešenje 5.13 Turistička agencija

```
import java.util.concurrent._
import java.util.Scanner
import java.io.File

class Ucesnik(ime : String,
              cena : Int,
```

```
              dobitnici : Array[String])
              extends Thread {

  override def run() {
//  Cekamo dok se ne zavrsi izvlacenje.
//
//  Metod wait() suspenduje nit, oslobadja kriticnu sekciju obmotanu synchronized blokom
//  i dozvoljava drugim nitima koje su zakljucale isti objekat da udju u kriticnu sekciju
//  sve dok neka druga nit ne pozove nad istim objektom metod notifyAll()
//  cime se obavestavaju sve niti koje cekaju sa metodom wait()
//  da mogu nastaviti sa radom
//
    dobitnici.synchronized {
      dobitnici.wait()
    }
    for(d <- dobitnici)
      if(d == ime){
        println("Cestitamo " + ime +
                "!!! Osvojili ste popust od 20% na cenu karte. Vasa karta sada kosta " + cena*0.8 + "e.")
        return
      }
    println("Nazalost " + ime +
            " niste osvojili popust, vise srece drugi put. Vasa karta kosta " + cena + "e.")

  }

  def getIme() : String = {
    return ime
  }
}

object TuristickaAgencija {
  def main(args : Array[String]) {
     val sc : Scanner = new Scanner(new File("ucesnici.txt"))

     val dobitnici = new Array[String](5)
     val n = sc.nextInt()
     sc.nextLine()
     val ucesnici = new Array[Ucesnik](n)
     for(i <- 0 until n){
        ucesnici(i) = new Ucesnik(sc.nextLine(), sc.nextInt(), dobitnici)
        sc.nextLine()
     }

     for(u <- ucesnici)
       u.start()

     println("Izvlacenje je u toku.")
     Thread.sleep(5000)
//   Ulazimo u kriticnu sekciju, i racunamo dobitnike nagradnih popusta
     dobitnici.synchronized {
       val izvuceniIndeksi = ThreadLocalRandom.current().ints(0, n).distinct().limit(5).toArray()
       for(j <- 0 until izvuceniIndeksi.length)
         dobitnici(j) = ucesnici(izvuceniIndeksi(j)).getIme()

//     Kada su izracunati dobitnici, obavestavamo niti koje cekaju
//     i izlazimo iz kriticne sekcije
       dobitnici.notifyAll()
     }

  }
}
```

6

Distribuirano programiranje

6.1 O Apache Spark-u

Apache Spark je radni okvir (eng. framework) za distribuirano programiranje. Pruža interfejse za programiranje (eng. API) u jezicima Java, Scala, Python i R.

Svaka Spark aplikacija se sastoji od glavnog (eng. *driver*) programa koji pokreće funkciju `main` i izvršava paralelne operacije na povezanom klaster računaru. Pristupanje klaster računaru se vrši pomoću objekta kontekst tipa `SparkContext`. Prilikom konstrukcije kontekst objekta, potrebno je definisati koji klaster računar će se koristiti. Spark aplikacija može koristiti lokalni računar kao simulaciju klaster računara (svaka procesorska jedinica će simulirati jedan čvor klaster računara) ili neki udaljeni klaster računar.

Spark koristi posebne kolekcije podataka koje se mogu obrađivati paralelno (eng. *RDD - Resilient Distributed Datasets*). Paralelne kolekcije se mogu napraviti od već postojećih kolekcija u programu ili se mogu učitati iz spoljašnjeg sveta. Nad paralelnim kolekcijama možemo izvršavati dva tipa operacija *transformacije* i *akcije* (slika 6.1). Transformacije transformišu kolekciju na klaster računaru i prave novu paralelnu kolekciju. Sve transformacije su lenje, što znači da rezultat izračunavaju u trenutku kada on postane potreban.

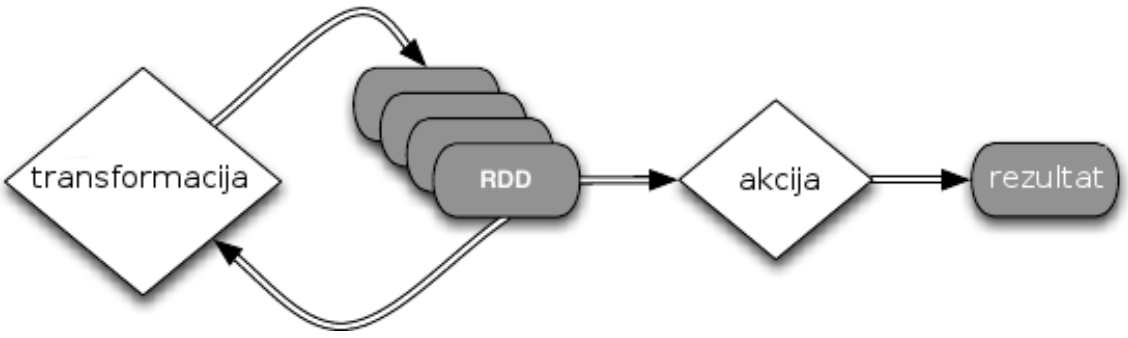
Akcije su operacije koje se izvršavaju na klaster računaru nad paralelnim kolekcijama i njihov rezultat (izračunata vrednost) se vraća na lokalni računar.

Spark može da sačuva paralelne kolekcije u memoriji čvorova klaster računara i na taj način ubrzati izvršavanje narednih operacija.

Spark pruža mogućnost korišćenja deljenih podataka u vidu emitovanih promenljivih (eng. *broadcast variables*) i akumulatora (eng. *accumulators*).

Neke od funkcija transformacija su:

- `map(f)` - vraća novu kolekciju koja se dobija tako što se primeni funkcija f nad svakim elementom postojeće kolekcije
- `filter(f)` - primenjuje funkciju f nad svim elementima kolekcije i vraća novu kolekciju koja sadrži one elemente za koje je funkcija f vratila true



Slika 6.1: *Operacije nad paralelnim podacima*

- `flatMap(f)` - slična je funkciji map, razlika je to što primena funkcije f nad nekim elementom kolekcije može da vrati 0 ili više novih elemenata koji se smeštaju u rezultujuću kolekciju
- `groupByKey()` - poziva se nad kolekcijom parova (kljuc, vrednost) i vraća kolekciju parova (kljuc, Iterable<vrednost>) tako što grupiše sve vrednosti sa istim ključem i smešta ih u drugi element rezultujućeg para
- `reduceByKey(f)` - poziva se nad kolekcijom parova (kljuc, vrednost) i vraća kolekciju parova (kljuc, nova_vrednost) , nova_vrednost se dobija agregiranjem svih vrednosti sa istim ključem koristeću zadatu funkciju agregacije f
- `aggregateByKey(pocetna_vrednost)(f1, f2)` - poziva se nad kolekcijom parova (kljuc, vrednost) i vraća kolekciju parova (kljuc, nova_vrednost) , nova_vrednost se dobija agregiranjem pocetne vrednosti i svih vrednosti sa istim ključem koristeću zadatu funkciju agregacije f1 u svakom čvoru klaster računara, a funkcija f2 agregira vrednosti izračunate u čvorovima klaster računara u jednu vrednost - nova_vrednost
- `sortByKey()` - poziva se nad kolekcijom parova (kljuc, vrednost) i vraća novu kolekciju sortiranu po ključu
- `cartesian(druga_kolekcija)` - spaja kolekciju sa drugom kolekcijom i vraća kolekciju svih parova (vrednost_iz_prve_kolekcije,vrednost_iz_druge_kolekcije)
- `zip(druga_kolekcija)` - spaja kolekciju sa drugom kolekcijom spajajući elemente na istim pozicijama i vraća kolekciju parova (vrednost_iz_prve_kolekcije, vrednost_iz_druge_kolekcije)

Neke od funkcija akcija su:

- `reduce(f)` - agregira elemente kolekcije koristeći funkciju f i vraća rezultat agregacije
- `collect()` - pretvara paralelnu kolekciju u niz (koji se nalazi na lokalnom računaru)
- `count()` - vraća broj elemenata kolekcije
- `countByKey()` - poziva se nad kolekcijom parova (kljuc, vrednost), za svaki ključ broji koliko ima elemenata sa tim ključem i vraća neparalelnu kolekciju (kljuc, broj_elemenata)
- `first()` - vraća prvi element kolekcije
- `take(n)` - vraća prvih n elemenata kolekcije
- `takeSample(sa_vracanjem, n, seed)` - vraća prvih n nasumično izabranih elemenata kolekcije (sa ili bez vraćanja), seed predstavlja početnu vrednost generatora slučajnih brojeva
- `takeOrdered(n[, poredak])` - vraća prvih n elemenata sortirane kolekcije (koristeći prirodan poredak kolekcije ili zadati poredak)
- `saveAsTextFile(ime_direktorijuma)` - upisuje kolekciju u datoteke koje se nalaze u zadatom direktorijumu
- `foreach(f)` - poziva funkciju f nad svim elementima kolekcije (uglavnom se koristi kada funkcija f ima neke sporedne efekte kao što je upisivanja podataka u datoteku ili slično)

Konfiguraciju Spark aplikacije možemo podešavati dinamički prilikom pokretanja aplikacije na klaster računaru. Potrebno je upakovati aplikaciju zajedno sa svim njenim bibliotekama u .jar datoteku koristeći neki od alata (Maven [1], SBT [2] i sl.) i instalirati Spark upravljač na klaster računaru (Standalone [3], Mesos [4], Yarn [5] i sl.). Aplikaciju možemo pokrenuti pomoci `spark-submit` skripta koji se nalazi u `bin` direktorijumu instaliranog Spark alata i konfigurisati dinamički. Na primer:

[1] http://maven.apache.org/
[2] http://www.scala-sbt.org/
[3] http://spark.apache.org/docs/latest/spark-standalone.html
[4] http://spark.apache.org/docs/latest/running-on-mesos.html
[5] http://spark.apache.org/docs/latest/running-on-yarn.html

```
./bin/spark-submit --class Main --master local --num-executors 20
   Aplikacija.jar
```

Parametri koji se najčešće koriste prilikom konfiguracije su:

- `--master url` - URL klaster racunara
- `--class ime_klase` - glavna klasa naše aplikacije
- `--num-executors n` - broj čvorova koji izvršavaju našu aplikaciju (eng. executors)
- `--executor-cores n` - broj zadataka koje jedan čvor može izvršavati istovremeno
- `--executor-memory n` - veličina hip memorije svakog čvora

6.2 Uputstvo za Apache Spark

Uputstvo prikazuje kako konfigurisati projekat u okruženju `Intellij Idea` da koristi biblioteku `Apache Spark`.

Literatura:

- spark.apache.org/docs/0.9.1/scala-programming-guide.html

6.2.1 Potreban softver i alati

Potrebno instalirati:

- Intellij Idea
  (jetbrains.com/idea/)
- Java JDK8
  (oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
- Plugin `Scala` za `Intellij Idea` (instalacija iz okruženja)

6.2.2 Pravljenje projekta

Koristi se `sbt` (eng. Simple build tool) koji će da preuzima i konfiguriše Spark biblioteku za nas. Na slici 6.2 je prikazano kako u okruženju odabrati `sbt` projekat.

Za pravljenje `sbt` projekta **neophodno** je imati aktivnu internet konekciju.

6.2.3 Konfiguracija projekta

Odaberite gde želite da se napravi projekat. Preporučeno je da imate direktorijum u kojem čuvate sve `Intellij Idea` projekte (uglavnom je to `home/korisnik/IdeaProjects`).

Za `sbt` i jezik `Scala` odaberite:

- `sbt: 0.13.max (0.13.17)`
- `Scala: 2.10.max (2.10.7)`

Na slici 6.3 prikazano je gde treba odabrati verzije za `sbt` i jezik `Scala`.

Kliknite `Finish`.

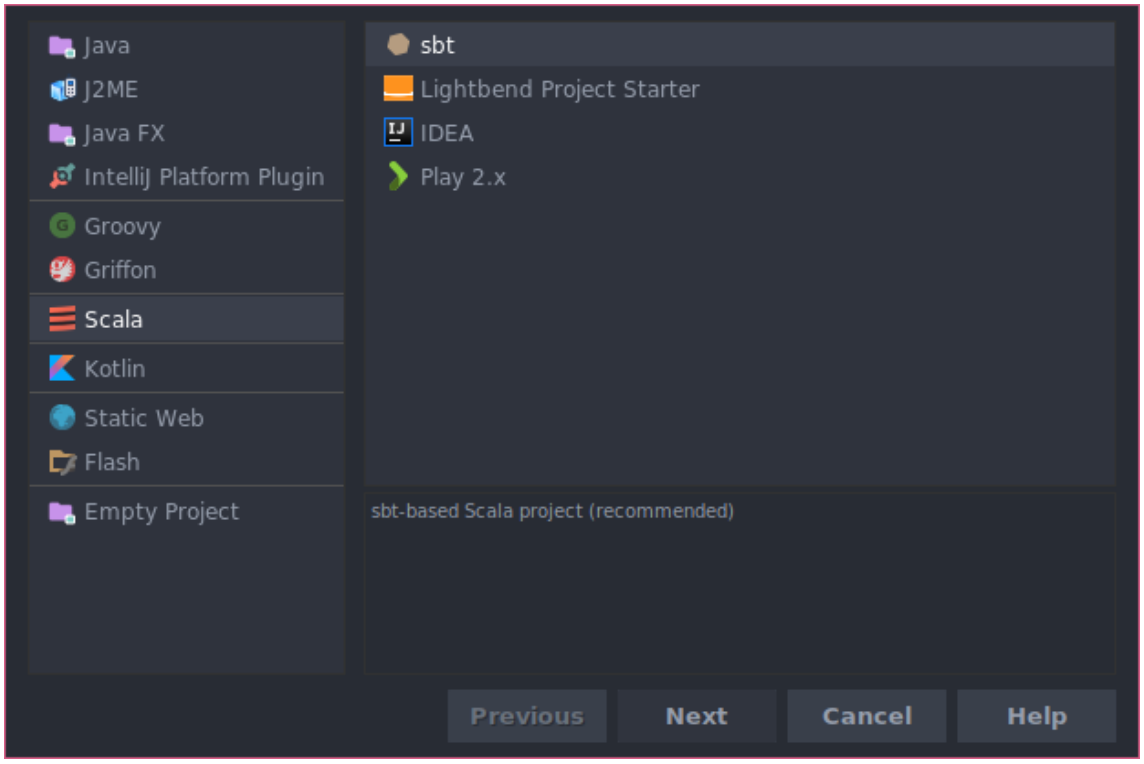
6.2.4 Inicijalizacija projekta

Okruženje se sada inicijalizuje, detektuju se Skala biblioteke i slično. Ovaj proces može da potraje od nekoliko sekundi do nekoliko minuta u zavisnosti od brzine mrežne konekcije i hardvera računara te treba biti strpljiv.

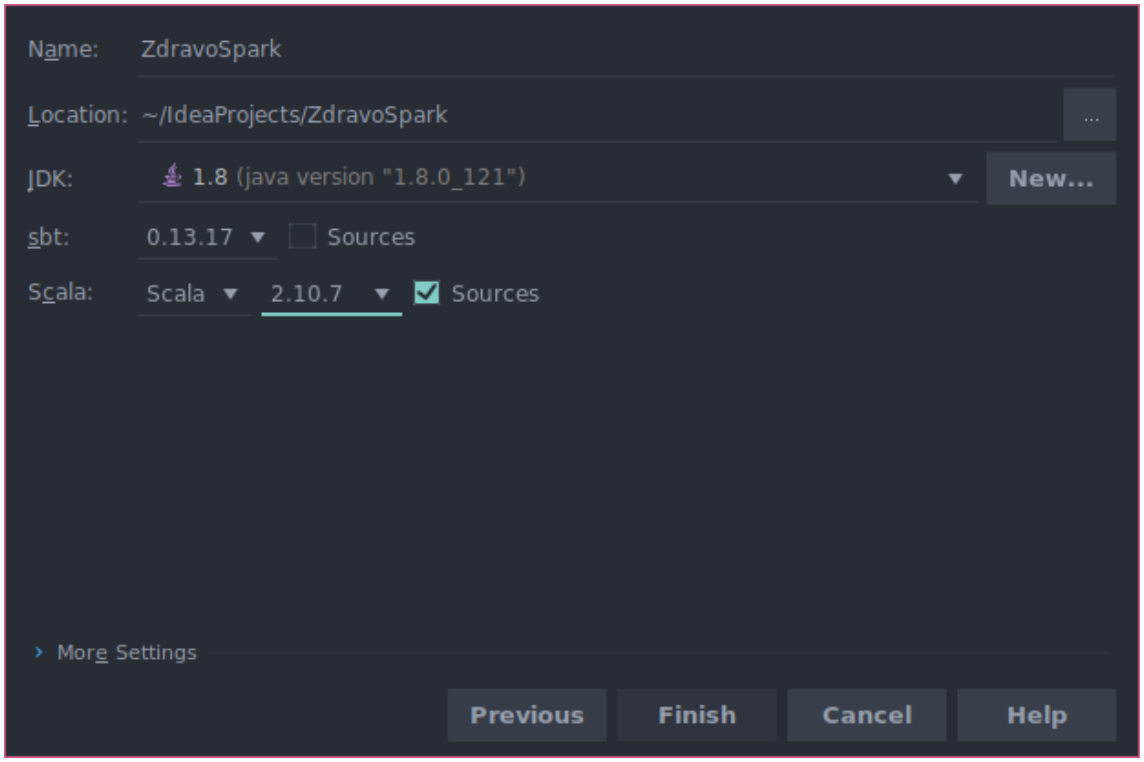
U donjem delu okruženja na sredini možete čitati poruke o tome šta se trenutno dešava, na primer:

`sbt: dump project structure from sbt`

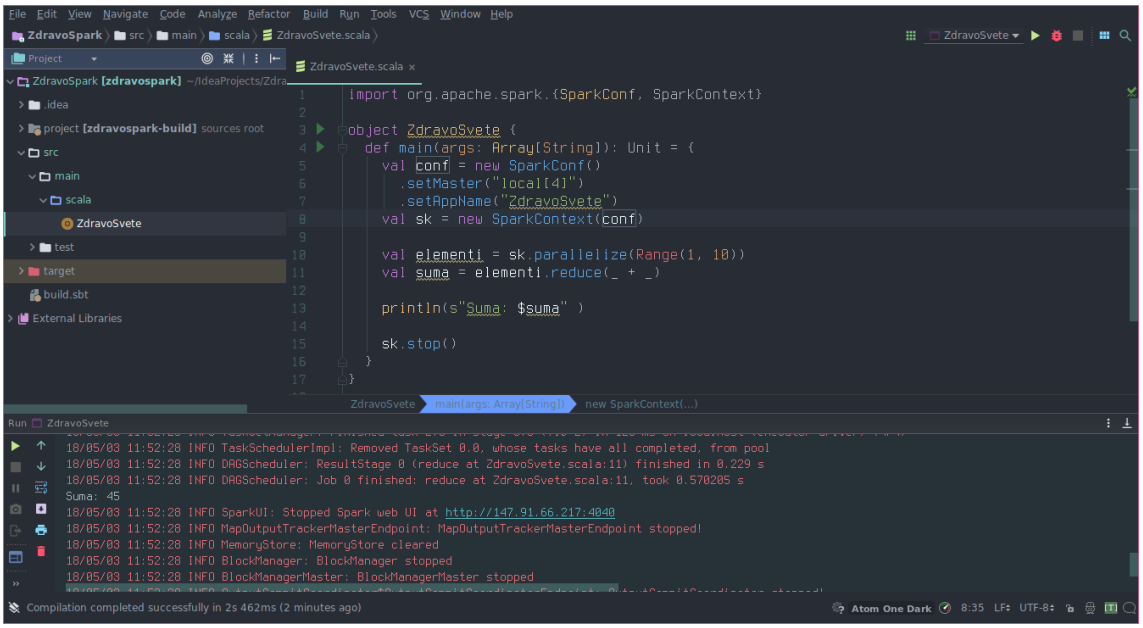
je jedan od koraka koje Intellij mora da izvrši kako bi pripremio sve za rad.



Slika 6.2: *Početak pravljenja* `sbt` *projekta*



Slika 6.3: *Kraj pravljenja* `sbt` *projekta*



Slika 6.4: *Pokretanje Spark programa*

6.2.5 Dodavanje biblioteke u `build.sbt`

Potrebno je da sistemu `sbt` definišemo da naš projekat koristi spoljnu biblioteku. U datoteci `build.sbt` možemo definisati zavisnost (eng. *dependency*) od spoljne biblioteke.

Potrebno je dodati sledeći kod u datoteku `build.sbt`

```
libraryDependencies ++= {
  val sparkVer = "2.1.0"
  Seq(
    "org.apache.spark" %% "spark-core" % sparkVer
  )
}
```

Vaš `build.sbt` bi trebao imati sledeći oblik:

```
name := "ZdravoSpark"
version := "0.1"
scalaVersion := "2.10.7"
libraryDependencies ++= {
  val sparkVer = "2.1.0"
  Seq(
    "org.apache.spark" %% "spark-core" % sparkVer
  )
}
```

Pri čemu atribut `name` zavisi od imena projekta koje ste originalno odabrali.

Sačuvajte izmene. Kada Vas okruženje pita da ažurira projekat jer je izmenjena datoteka `build.sbt` prihvatite izmene. Alat `sbt` će u skladu sa Vašim izmenama preduzeti odgovarajuće akcije. U ovom slučaju to će biti preuzimanje biblioteke `spark-core` sa odgovarajućeg repozitorijuma i njeno uključivanje u Vaš projekat.

6.2.6 Pokretanje programa

Nakon što je sbt pripremio okruženje za rad, možemo da pristupimo Spark biblioteci. Na slici 6.4 je prikazano pokretanje spark programa.

6.2.7 Zadaci sa rešenjima

Zadatak 6.1 **Parni kvadrati** Napisati program koji učitava ceo broj n veći od 2 i ispisuje sve kvadrate parnih brojeva počev od broja 2 do n.

[Rešenje 6.1]

Zadatak 6.2 **Broj petocifrenih** Napisati program koji ispisuje broj petocifrenih brojeva koji se nalaze u datoteci *brojevi.txt* (svaka linija sadrzi jedan broj).

[Rešenje 6.2]

Zadatak 6.3 **Skalarni proizvod** Napisati program koji racuna skalarni proizvod dva vektora (pretpostavimo da su vektori uvek zadati ispravno tj. iste su dužine) i ispisuje ih na izlaz. Vektori se nalaze u datotekama *vektor1.txt* i *vektor2.txt* u formatu:

```
1 a1, a2, a3, a4, ... an
```

[Rešenje 6.3]

Zadatak 6.4 **Broj pojavljivanja reči** Napisati program koji za svaku reč iz knjige (datoteka *knjiga.txt*) broji koliko se puta ona pojavljuje i rezultat upisuje u datoteku.

[Rešenje 6.4]

Zadatak 6.5 **Uređaji transakcije** U datoteci *uredjaji.txt* se nalaze podaci o kupljenim uređajima u formatu:

```
1 marka_uredjaja ostali_podaci
```

Napisati program koji izdvaja podatke o svim transakcijama jedne marke i upisuje ih u posebnu datoteku sa nazivom *ime_marke.txt*.

[Rešenje 6.5]

Zadatak 6.6 **Log poruke** Datoteka *log.txt* sadrži podatke koji su generisani pokretanjem Java programa. Napisati program koji izdvaja poruke koje se odnose na pakete jezika Java grupisane po tipu poruke i ispisuje ih na izlaz.
Poruke mogu biti informacione, upozorenja ili greške.
Format poruka je:

```
1 tip ostatak_poruke
```

Tip moze biti `[warn]`, `[info]` ili `[error]`.

[Rešenje 6.6]

Zadatak 6.7 **Uspešna preuzimanja** U datoteci *mavenLog.txt* se nalaze podaci o započetim/uspešnim preuzimanjima paketa prilikom pokretanja Maven alata (upravljač zavisnostima) u formatu:

```
1  Downloading: ostatak_poruke
```

ili

```
1  Downloaded: ostatak_poruke
```

Napisati program koji računa procenat uspešnih preuzimanja paketa u odnosu na započeta preuzimanja.

[Rešenje 6.7]

Zadatak 6.8 **Pokloni** Bliži se Božić i firma želi da pokloni svojim zaposlenim programerima tri paketića. Sin direktora firme je budući programer i želi da napravi program koji će simulirati izvlačenje troje dobitnika paketića. Kako još uvek nije dobro savladao programiranje, pomozimo mu tako što ćemo napisati program koji nasumicno bira tri programera i ispisuje njihova imena, prezimena i email.
Podaci o zaposlenima se nalaze u datoteci *zaposleni.txt* u formatu:

```
ime prezime pol identifikator IP_adresa_racunara datum_zaposlenja
  sifra_pozicije plata
```

Šifra pozicije programera je `IT_PROG`.

[Rešenje 6.8]

Zadatak 6.9 **Prosečna temperatura** U datoteci *temperaturaBoston.txt* se nalaze podaci o prosečnim temperaturama u Bostonu od 1995 do 2016 godine u Farenhajtima. Napisati program koji ispisuje prosečne temperature u Bostonu za svaku godinu od 1995 do 2016 godine posebno u Celzijusima.
Format podataka je:

```
 mesec dan godina temperatura
```

[Rešenje 6.9]

6.2.8 Zadaci za vežbu

Zadatak 6.10 Napisati program koji učitava ceo broj n i ispisuje faktorijele svih brojeva od 1 do n.

Zadatak 6.11 Napisati program koji učitava ceo broj n i ispisuje sumu prvih n elemenata harmonijskog reda (podsetimo se, harmonijski red je red oblika 1 + 1/2 + 1/3 + 1/4 ...).

Zadatak 6.12 Napisati program koji racuna zbir dva vektora (pretpostavimo da su vektori uvek zadati ispravno tj. iste su dužine) i ispisuje ih na izlaz.
Vektori se nalaze u datotekama *vektor1.txt* i *vektor2.txt* u formatu:

```
a1, a2, a3, a4, ... an
```

Zadatak 6.13 Napisati program koji racuna broj pojavljivanja svake cifre 0-9 u datoteci *knjiga.txt* i ispisuje rezultat sortiran po ciframa.

Zadatak 6.14 Napisati program koji prebrojava sve poruke o greškama koje se odnose na Spark alat i rezultat ispisuje na izlaz. Poruke o greeskama se nalaze u datoteci *log.txt* u formatu:

```
tip ostatak_poruke
```

Tip poruke o grešci je `[error]`.

Zadatak 6.15 Napisati program koji izdvaja dane sa najvišom temperaturom u Bostonu za svaku godinu posebno, počev od 1995 do 2016 i rezultat upisuje u direktorijum *MaxTemp*. Podaci o temperaturama se nalaze u datoteci *temperaturaBoston.txt* u formatu:

```
mesec dan godina temperatura
```

Zadatak 6.16 Napisati program koji izdvaja podatke o veličini preuzetih paketa koji se odnose na Apache server i rezultat upisuje u direktorijum *ApacheDownloaded*. Podaci o preuzimanjima se nalaze u datoteci *mavenLog.txt* u formatu:

```
Downloaded: putanja podaci_o_velicini
```

Zadatak 6.17 Ekonomski analitičari žele da analiziraju tržište tehničke robe sa nekim rasponom cena. Napisati program koji učitava ime marke i bira nasumično pet poruka o transakcijama koje se odnose na tu marku. Podaci o transakcijama se nalaze u datoteci *uredjaji.txt* u formatu:

```
marka_uredjaja ostali_podaci
```

Zadatak 6.18 Napisati program koji računa prosečnu platu programera u firmi. Podaci o zaposlenima se nalaze u datoteci *zaposleni.txt* u formatu:

```
ime prezime pol identifikator IP_adresa_racunara datum_zaposlenja
   sifra_pozicije plata
```

Šifra pozicije programera je `IT_PROG`.

6.3 Rešenja

Rešenje 6.1 Parni kvadrati

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD._

object ParniKvadrati {

  def main(args: Array[String]) = {

    println("Unesite broj n:")
    val n = Console.readInt()
    /**
     * Podesavamo konfiguraciju Spark okruzenja
     * tako sto dajemo ime aplikaciji
     * i dodeljujemo joj potencijalno 4 cvora
     * (u nasem slucaju procesorska jezgra)
     * */
    val konf = new SparkConf()
      .setAppName("ParniKvadrati")
      .setMaster("local[4]")

    /**
     * Pravimo objekat Spark konteksta
     * koji pokrece i upravlja Spark okruzenjem
     * */
    val sk = new SparkContext(konf)


    /**
     * Ukoliko zelimo da podesavamo parametre dinamicki
     * (broj cvorova koji izvrsavaju nasu aplikaciju,
     * velicina hip memorije i sl.)
     * potrebno je da inicijalizujemo spark kontekst na sledeci nacin
     *
     * val sk = new SparkContext(new SparkConf());
     *
     * cime naznacavamo da ce se parametri konfiguracije
     * podesiti dinamicki (koriscenjem spark-submit skripte).
     * */

    val niz = (2 to n by 2).toArray

    /**
     * Pravimo niz tipa RDD[Integer] od niza tipa Array[Integer]
     * */
    val nizRDD = sk.parallelize(niz)

    /**
     * Pravimo niz kvadrata parnih brojeva (uzimamo prvih 10)
     * i rezultat pretvaramo u niz tipa Array[Integer]
     * */
```

```
51      val nizKvadrata = nizRDD.map(x => x*x)
                              /* .take(10) */
53                               .collect()

55      /**
         * Zaustavljamo Spark okruzenje
57       * */
        sk.stop()
59
        /**
61       * Ispisujemo rezultujuci niz
         * */
63      println("Niz kvadrata parnih brojeva: ")
        println(nizKvadrata.mkString(", "))
65    }
}
```

Rešenje 6.2 Broj petocifrenih

```
import org.apache.spark.SparkConf
2 import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD._
4

6 object BrojPetocifrenih {

8   def main(args: Array[String]){

10      val konf = new SparkConf()
          .setAppName("BrojPetocifrenih")
12        .setMaster("local[4]")

14      val sk = new SparkContext(konf)

16      /**
         * Otvaramo datoteku i njen sadrzaj cuvamo
18       * u nizu tipa RDD[String].
         * Elementi niza su pojedinacne linije iz datoteke.
20       * */
        val datRDD = sk.textFile("brojevi.txt")
22
        /**
24       * Filtriramo niz tako da nam ostanu samo petocifreni brojevi
         * i prebrojavamo ih.
26       * */
        val brojPetocifrenihBrojeva = datRDD.filter(_.length() == 5)
28                                          .count()

30      sk.stop()

32      println("Petocifrenih brojeva ima: ")
        println(brojPetocifrenihBrojeva)
34    }
}
```

Rešenje 6.3 Skalarni proizvod

```
1 import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
3 import org.apache.spark.rdd.RDD._

5 object SkalarniProizvod {

7    def main(args: Array[String]){

9       val konf = new SparkConf()
          .setAppName("SkalarniProizvod")
11        .setMaster("local[4]")

13      val sk = new SparkContext(konf)
```

```
15        /**
17         * Otvaramo datoteku i ucitavamo vektor.
           * */
19        val vek1RDD = sk.textFile("vektor1.txt")
                        /**
21                      * Razdvajamo elemente vektora iz niske koristeci separator ", "
                        * */
23                      .flatMap(_.split(", "))
                        /**
25                      * Kastujemo niske u tip Integer.
                        * */
27                      .map(_.toInt)

29        val vek2RDD = sk.textFile("vektor2.txt")
                        .flatMap(_.split(", "))
31                      .map(_.toInt)

33        /**
           * Spajamo nizove vektora A i B funkcijom zip
35         * i pravimo jedan niz parova (tipa Tuple)
           * tako da svaki par sadrzi element vektora A i element vektora B
37         *  (a1,b1), (a2,b2), ... (an, bn).
           * */
39        val skProizvod = vek1RDD.zip(vek2RDD)
                               /**
41                             * Mnozimo elemente para.
                               * */
43                             .map(par => par._1 * par._2)
                               /**
45                             * Pomnozene elemente parova sabiramo.
                               * */
47                             .reduce((a, b) => a+b)
          sk.stop()

49        println("Skalarni proizvod je: ")
          println(skProizvod)
51    }
  }
```

Rešenje 6.4 Broj pojavljivanja reči

```
import org.apache.spark.SparkConf
2 import org.apache.spark.SparkContext
  import org.apache.spark.rdd.RDD._
4
  object BrojPojavljivanjaReci {
6
     def main(args: Array[String]){
8
        val konf = new SparkConf()
10        .setAppName("BrojPojavljivanjaReci")
          .setMaster("local[4]")
12
        val sk = new SparkContext(konf)
14
        val knjigaRDD = sk.textFile("knjiga.txt")
16
        /**
18       * Ucitavamo linije i razlazemo ih separatorom " " tako da dobjemo niz reci
         */
20      val reciBr = knjigaRDD.flatMap(_.split(" "))
                              /**
22                            * Od svake reci pravimo par (rec, 1).
                              */
24                            .map(rec => (rec , 1))
                              /**
26                            * Sabiramo sve vrednosti drugog elementa para
                              * grupisane po prvom elementu para
28                            * koji nam predstavlja kljuc.
                              */
30                            .reduceByKey((_+_))
```

```
                            /**
32                          * Sortiramo reci leksikografski.
                            */
34                          .sortByKey()
                            /**
36                          * Cuvamo ih u datotekama koje se nalaze u direktorijumu
                            * BrojPojavljivanjaReci
38                          */
                            .saveAsTextFile("BrojPojavljivanjaReci")
40
      sk.stop()
42  }
}
```

Rešenje 6.5 Uređaji transakcije

```
1  import org.apache.spark.SparkConf
   import org.apache.spark.SparkContext
3  import org.apache.spark.rdd.RDD._
   import java.io._
5
   object UredjajiTransakcije {
7
     def main(args: Array[String]){
9
       val konf = new SparkConf()
11       .setAppName("UredjajiTransakcije")
         .setMaster("local[4]")
13
       val sk = new SparkContext(konf)
15
       val transakcije = sk.textFile("uredjaji.txt")
17                       /**
                          * Razdvajamo podatke o uredjajima
19                        * i pravimo parove transakcija
                          * (marka, ostali_podaci)
21                        * */
                         .map(linija => {
23                           val niz = linija.split(" ")
                             (niz(0), niz.drop(1).mkString(" "))
25                         })
                         /**
27                       * Grupisemo ih po marki tako da za svaku marku
                         * cuvamo niz obavljenih transakcija
29                       * (tj. niz koji sadrzi ostale podatke za svaku transakciju)
                         * */
31                       .groupByKey()
                         /**
33                        * Prolazimo kroz niz parova (marka, niz_transakcija)
                          * i u datoteku ime_marke.txt
35                        * upisujemo podatke o transakcijama.
                          *
37                        * Parametar t u foreach konstrukciji predstavlja jedan par
                          * (marka, niz_transakcija).
39                        * */
                         .foreach(t => {
41                           val dat = new PrintWriter(new File(t._1.toLowerCase() +
     ".txt" ))

43                           dat.write("---" + t._1 + "---\n")

45                           t._2.foreach(por => {
                                 dat.append(por + "\n")
47                           })

49                           dat.close()
                         })
51
       sk.stop()
53   }
}
```

Rešenje 6.6 Log poruke

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD._

object LogPoruke {

  def main(args: Array[String]){

      val konf = new SparkConf()
        .setAppName("LogPoruke")
        .setMaster("local[4]")

      val sk = new SparkContext(konf)

      val poruke = sk.textFile("log.txt")
                     /**
                      * Filtriramo podatke tako da nam ostanu
                      * samo linije koje predstavljaju
                      * upozorenja, informacije ili greske i odnose se na javu.
                      * */
                     .filter(linija =>
                                (linija.contains("[warn]")
                                 || linija.contains("[info]")
                                 || linija.contains("[error]"))
                                && (linija.contains("java")))
                     /**
                      * Pravimo parove (tip_poruke, poruka).
                      * */
                     .map(linija => {
                          val niz = linija.split(" ")
                          (niz(0), niz.drop(1).mkString(" "))
                     })
                     /**
                      * Grupisemo poruke po njihovom tipu (kljuc).
                      * tako da dobijemo niz parova (tip_poruke, niz_poruka).
                      * */
                     .groupByKey()
                     /**
                      * Za svaki tip racunamo broj poruka tog tipa
                      * tako sto od parova (tip_poruke, niz_poruka)
                      * pravimo par (tip_poruke, broj_poruka)
                      * */
                     .map( por => (por._1, por._2.size))
                     .collect()

      println("Informacije o log porukama koje se odnose na Javu: ")

      poruke.foreach( por => println(" " + por._1 + ": " + por._2 ))

      sk.stop()
    }
}
```

Rešenje 6.7 Uspešna preuzimanja

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD._

object UspesnaPreuzimanja {

  def main(args: Array[String]){

      val konf = new SparkConf()
        .setAppName("UspesnaPreuzimanja")
        .setMaster("local[4]")

      val sk = new SparkContext(konf)

      /**
```

```
        * Ucitavamo podatke i smestamo ih u kes memoriju radi brzeg pristupanja.
        * */
       val preuzimanja = sk.textFile("mavenLog.txt")
                           .cache()
       /**
        * Racunamo broj zapocetih preuzimanja.
        * */
       val zapoceta = preuzimanja.filter(_.contains("Downloading:"))
                                 .count()
       /**
        * Racunamo broj zavrsenih preuzimanja.
        * */
       val zavrsena = preuzimanja.filter(_.contains("Downloaded:"))
                                 .count()

       sk.stop()

       println("%.2f".format(zavrsena*100.0/zapoceta) + " procenata zapocetih
     preuzimanja je zavrseno.")
    }
}
```

Rešenje 6.8 Pokloni

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD._
import scala.compat._

object RandomZaposleni {

  def main(args: Array[String]){

       val konf = new SparkConf()
         .setAppName("RandomZaposleni")
         .setMaster("local[4]")

       val sk = new SparkContext(konf)

       /**
        * Pronalazimo liniju koja sadrzi radnika programera,
        * razdvajamo podatke o jednom radniku
        * i cuvamo njegovo ime, prezime i email nalog.
        * Nakon toga nasumicno biramo 3 programera.
        *
        * Funkcija takeSample kao argumente prihvata:
        * - indikator da li zelimo izbora sa vracanjem
        * - broj uzoraka
        * - pocetnu vrednost (seed) za slucajni generator
        * */
       val triProgramera = sk.textFile("zaposleni.txt")
                             /**
                              * Pronalazimo liniju koja sadrzi radnika programera,
                              * */
                             .filter(_.contains("IT_PROG"))
                             /**
                             * Razdvajamo podatke o jednom radniku
                             * i cuvamo njegovo ime, prezime i email nalog.
                             * */
                             .map(linija => {
                                 val niz = linija.split(" ")
                                 (niz(0), niz(1), niz(3))
                               })
                             /**
                             * Nasumicno biramo 3 programera.
                             * Prvi parametar (false) oznacava
                             * izbor bez vracanja a poslednji parametar
                             * predstavlja pocetnu vrednost (seed)
                             * generatora slucajnih brojeva.
                             * */
                             .takeSample(false, 3, Platform.currentTime)
```

```
49        println("Tri zaposlena radnika u IT sektoru su: ")
          triProgramera.foreach(prog => {
51          println("Ime i prezime: " + prog._1 + " " + prog._2 + "\n  Email: " + prog._3
      .toLowerCase()+"@firma.com")
          })
53
          sk.stop()
55    }
}
```

Rešenje 6.9 Prosečna temperatura

```
1  import org.apache.spark.SparkConf
   import org.apache.spark.SparkContext
3  import org.apache.spark.rdd.RDD._

5  object ProsecnaTemperatura {

7     def main(args: Array[String]){

9        val konf = new SparkConf()
           .setAppName("ProsecnaTemperatura")
11         .setMaster("local[4]")

13       val sk = new SparkContext(konf)

15       val tempRDD = sk.textFile("temperatureBoston.txt")
                        /**
17                       * Pravimo torke (kljuc, vrednost) takve da je
                         * kljuc = godina
19                       * vrednost = (mesec, dan, temperatura)
                         * */
21                      .map(linija => {
                          val niz = linija.split(" ")
23                        (niz(3),(niz(1), niz(2), niz(4).toFloat))
                        })
25                      /**
                         * Grupisemo torke po njihovom kljucu i za svaki kljuc racunamo
27                       * sumu svih temperatura i broj temperatura koje smo sabrali.
                         *
29                       * Funkcija aggregateByKey(pocetnaVrednostAkumulatora)(f1, f2)
                         * obradjuje niz parova (kljuc, vrednost)
31                       * grupise ih po kljucu, akumulira vrednosti
                         * (inicijalna vrednost akumulatora
33                       * se prosledjuje kao parametar)
                         * i kao rezultat vraca niz parova (k, akumuliranaVrednost).
35                       *
                         * Funkcija f1(akumulator, vrednost) se primenjuje
37                       * nad svim vrednostima koje se nalaze u jednom cvoru
                         * i rezultat se smesta u akumulator tog cvora
39                       * Funkcija f2(akumulator1, akumulator2) se primenjuje
                         * nad svim izracunatim akumulatorima pojedinacnih cvorova
41                       * i rezultat se smesta u globalni akumumulator.
                         *
43                       * U nasem slucaju, akumulator ce da sadrzi dve vrednosti
                         * (sumaTemp, brojTemp) sa nulom kao pocetnom vrednoscu.
45                       * Funkcija f1(akumulator, vrednost)
                         * sabira vrednost (temperaturu)
47                       * sa sumaTemp iz akumulatora, a brojTemp povecava za 1.
                         * Funkcija f2(akumulator1, akumulator2)
49                       * sabira obe vrednosti ova dva akumulatora.
                         *
51                       * Kao rezultat primene ove funkcije dobicemo niz parova
                         * (kljuc, (sumaTemp, brojTemp))
53                       * gde nam je kljuc godina
                         * u kojoj zelimo da izracunamo prosecnu temperaturu.
55                       * */
                        .aggregateByKey((0.0, 0))((ak, vr) => (ak._1 + vr._3 , ak._2 +
    1 ),
57                                                (a1 ,a2) => (a1._1 + a2._1, a1._2 + a2.
    _2))
                        /**
```

```
59                          * Pravimo niz parova (kljuc, prosecnaTemperatura)
                            * */
61                         .map( st => (st._1, st._2._1/st._2._2))
                           .sortByKey()
63                         .collect()
                           .foreach( st => println("Godine "
65                                              + st._1
                                                + " prosecna temperatura je iznosila "
67                                              + "%.2f".format(((st._2-32)/1.8))
                                                + " celzijusa. "))
69
       sk.stop()
71   }
}
```

7

Komponentno programiranje

7.1 ScalaFX - Instalacija potrebnih paketa

Koristicemo jezik `Scala` (verzija 2.11) http://www.scala-lang.org/.
Potrebno je imati instalirano:

(a) `Intellij Idea` jetbrains.com/idea/

(b) Scala plugin za `Intellij Idea`

(c) SceneBuilder 2.0 (oracle.com/technetwork/java/javafxscenebuilder-1x-archive-2199384.html)

Potrebno je napraviti `sbt` projekat (videti uputstvo za Distribuirano programiranje u delu 6.2).
U datoteku `build.sbt` dodati:

```
libraryDependencies += "org.scalafx" %% "scalafx" % "8.0.144-R12"
```

Vaš `build.sbt` bi trebao izgledati:

```
name := "ZdravoScalaFX"
version := "0.1"
scalaVersion := "2.11.12"
libraryDependencies += "org.scalafx" %% "scalafx" % "8.0.144-R12"
```

Pri čemu vrednost za ključ `name` zavisi od imena projekta koje ste izabrali.
Literatura:

(a) http://www.scalafx.org/api/8.0/index.html#package

(b) https://github.com/scalafx/ScalaFX-Tutorials

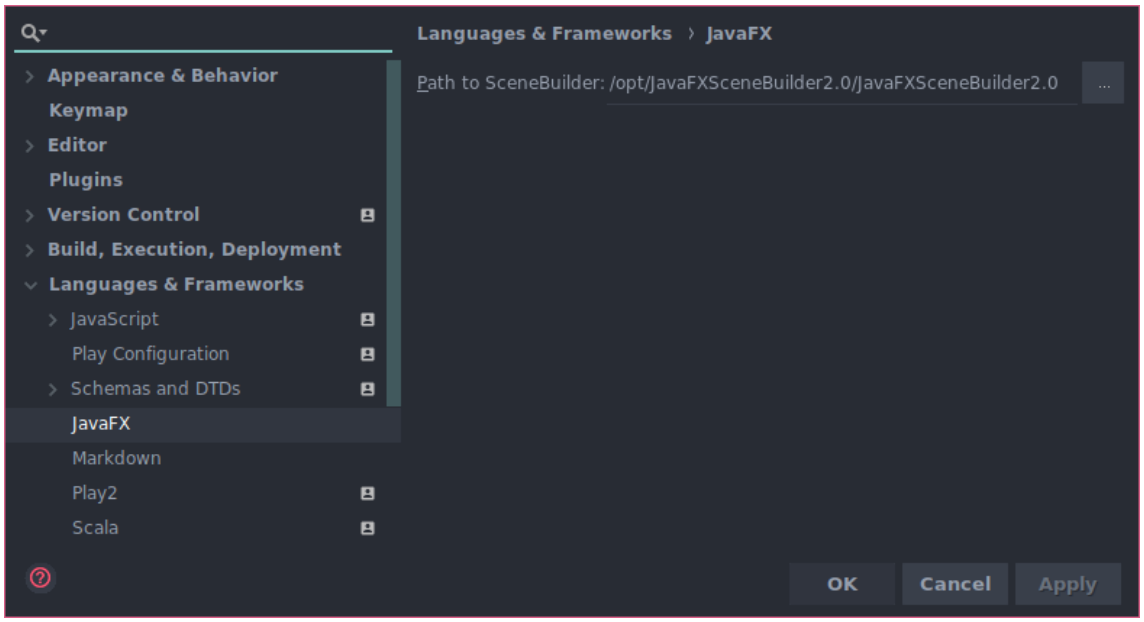
(c) http://www.oracle.com/technetwork/java/javafxscenebuilder-1x-archive-2199384.html

7.1.1 Instalacija paketa SceneBuilder

Preuzmite `Linux 32-bit (tar.gz)` ili `Linux 32-bit (tar.gz)` u zavisnosti od Vašeg sistema. Pretpostavimo da ste preuzeli 64bitnu verziju i smestili je u direktorijum `Downloads`. Izvršite sledeće:

```
cd ~/Downloads
tar -xvf  javafx_scenebuilder-2_0-linux-x64.tar.gz
sudo mv JavaFXSceneBuilder2.0 /opt
rm javafx_scenebuilder-2_0-linux-x64.tar.gz
```

U okruženju `Intellij Idea` je potrebno postaviti putanju to paketa `SceneBuider`. Kliknite na: `File -> Settings -> Languages and Frameworks -> JavaFX` i postavite putanju do paketa `SceneBuilder`. Prethodnim naredbama paket je postavljen na putanju `/opt/JavaFXSceneBuilder2.0` u kojoj se nalazi izvršiva datoteka `/opt/JavaFXSceneBuilder2.0/JavaFXSceneBuilder2.0`. Na slici 7.1 je ilustrovan proces odabira izvršive datoteke.



Slika 7.1: *Odabir putanje do paketa* `SceneBuilder`

7.1.2 Uvod

Zadatak 7.1 Napisati program koji korisniku omogućava da vrši konverziju iz dinara u evre i obratno. Za vrednost 1 evra uzeti 116.7 dinara, a ukoliko dođe do greške pri konverziji, obavestiti o tome korisnika (slika **??**). Koristeći alat `SceneBuilder` napraviti grafički korisnički interfejs kao na slikama **??** i **??**. Prikazani su kodovi redom za datoteke:

- `ConverterApp.scala`
- `ConverterController.scala`
- `ConverterLayout.fxml`

```
package example01

import java.io.IOException
import javafx.event.ActionEvent
import javafx.fxml.FXMLLoader
import javafx.scene.{Parent, Scene}

import scalafx.Includes._
import scalafx.application.JFXApp
import scalafx.application.JFXApp.PrimaryStage

object ConverterApp extends JFXApp {
  val resource = getClass.getResource("ConverterLayout.fxml")
  if (resource == null) {
    throw new IOException("Neuspelo ucitavanje resursa MoneyConverter.fxml")
  }

  val root: Parent = FXMLLoader.load(resource)

  stage = new PrimaryStage {
    title = "Dinar/Euro Converter"
    scene = new Scene(root)
  }
}
```

```
package example01

import java.net.URL
import java.util.ResourceBundle
```

```
import javafx.event.ActionEvent
import javafx.fxml.{FXML, Initializable}
import javafx.scene.control.{Button, TextField}
import javafx.scene.text.Text

class ConverterController extends Initializable {
  // Ime reference na objekat mora da se poklapa sa
  // atributom fx:id u okviru odgovarajuce .fxml datoteke.
  // Anotacija @FXML omogucava da se pri pokretanju programa
  // referenca poveze sa objektom koji nastaje na osnovu
  // .fxml datoteke.
  @FXML private var tfEuro: TextField = _
  @FXML private var tfDinar: TextField = _
  @FXML private var btEuroToDinar: Button = _
  @FXML private var btDinarToEuro: Button = _
  @FXML private var textMsg: Text = _

  val dinarValue = 116.7

  // Anotacija @FXML nam omogucava da se metod poveze
  // sa odgovarajucim javaFX dogadjajem.
  @FXML
  private def handleEuroToDinar(event: ActionEvent): Unit = {
    try {
      val dinarVal = tfEuro.getText.trim.toDouble * dinarValue
      tfDinar.setText(dinarVal.toString)
      textMsg.setText("")
    } catch {
      case _: Exception => textMsg.setText("Molimo unesite broj.")
    }
  }

  @FXML
  private def handleDinarToEuro(event: ActionEvent): Unit = {
    try {
      val euroVal = tfDinar.getText.trim.toDouble / dinarValue
      tfEuro.setText(euroVal.toString)
      textMsg.setText("")
    } catch {
      case _: Exception => textMsg.setText("Molimo unesite broj.")
    }
  }

  override def initialize(location: URL, resources: ResourceBundle): Unit = {}
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="
    -Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/8
    " xmlns:fx="http://javafx.com/fxml/1" fx:controller="example01.
    ConverterController">
  <columnConstraints>
    <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
    <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
      <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  </rowConstraints>
   <children>
      <Text fx:id="textEuro" strokeType="OUTSIDE" strokeWidth="0.0" text="Euro"
    GridPane.valignment="CENTER" />
```

Slika 7.2: *Konverzija valute*

Slika 7.3: *Neuspela konverzija*

```
      <Text fx:id="textDinar" strokeType="OUTSIDE" strokeWidth="0.0" text="Dinar"
    GridPane.rowIndex="1" />
      <TextField fx:id="tfEuro" GridPane.columnIndex="1" />
      <TextField fx:id="tfDinar" GridPane.columnIndex="1" GridPane.rowIndex="1" />
      <Button fx:id="btEuroToDinar" mnemonicParsing="false" onAction="#
    handleEuroToDinar" text="Euro -&gt; Dinar" GridPane.rowIndex="2" />
      <Button fx:id="btDinarToEuro" mnemonicParsing="false" onAction="#
    handleDinarToEuro" text="Dinar -&gt; Euro" GridPane.columnIndex="1" GridPane.
    rowIndex="2" />
      <Text fx:id="textMsg" fill="RED" strokeType="OUTSIDE" strokeWidth="0.0"
    GridPane.columnSpan="2" GridPane.halignment="CENTER" GridPane.rowIndex="3"
    GridPane.valignment="CENTER" />
   </children>
   <padding>
      <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
   </padding>
</GridPane>
```

Zadatak 7.2 Napisati program koji kolekciju podataka (lista pokemona) povezuje sa `JavaFX` komponentom `ListView` i omogućava njihovo leksikografsko sortiranje. Omogućiti korisniku da odabirom radio dugmeta odabere poredak sortiranja koje se potom može izvršiti na dugme `Sort`. Obezbediti da u svakom trenutku bude odabrano tačno jedno radio dugme.

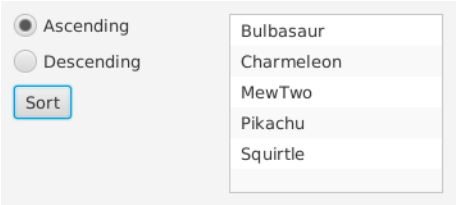
Koristeći alat `SceneBuilder` napraviti grafički korisnički interfejs kao na slikama ?? i ??. Prikazani su kodovi redom za datoteke:

- `PokemonApp.scala`
- `PokemonController.scala`
- `pokemonLayout.fxml`

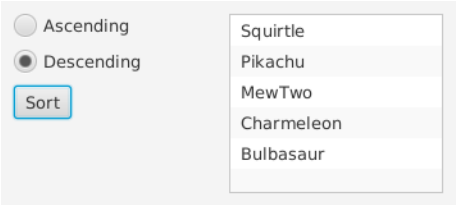
```
package example02

import java.io.IOException
import javafx.event.ActionEvent
import javafx.fxml.FXMLLoader
import javafx.scene.{Parent, Scene}

import scalafx.Includes._
import scalafx.application.JFXApp
import scalafx.application.JFXApp.PrimaryStage

object PokemonApp extends JFXApp {
  val resource = getClass.getResource("pokemonLayout.fxml")
  if (resource == null) {
    throw new IOException("Neuspelo ucitavanje resursa pokemonLayout.fxml")
  }

  val root: Parent = FXMLLoader.load(resource)

  stage = new PrimaryStage {
    title = "Pokemoni"
    scene = new Scene(root)
  }
}
```

```
package example02

import java.net.URL
import java.util.ResourceBundle
import javafx.collections.FXCollections
import javafx.event.ActionEvent
import javafx.fxml.{FXML, Initializable}
import javafx.scene.control.{Button, ListView, RadioButton, ToggleGroup}

import scalafx.collections.ObservableBuffer

class PokemonController extends Initializable {
  @FXML private var btSort: Button = _
  @FXML private var rbAsc: RadioButton = _
  @FXML private var rbDesc: RadioButton = _
  @FXML private var lvPokemons: ListView[String] = _

  private var tgRadios: ToggleGroup = new ToggleGroup()

  private var pokemons: ObservableBuffer[String] =
    new ObservableBuffer[String](
      FXCollections.observableArrayList(
        "Pikachu",
        "Charmeleon",
        "Bulbasaur",
        "Squirtle",
        "MewTwo"
      )
    )

  @FXML
  private def handleSort(event: ActionEvent): Unit = {
    if (rbAsc.isSelected) pokemons.sort(_ < _) else pokemons.sort(_ > _)
  }

  override def initialize(location: URL, resources: ResourceBundle): Unit = {
    rbAsc.setToggleGroup(tgRadios)
    rbDesc.setToggleGroup(tgRadios)

    // Povezumo komponentu 'ListView' tako da osluskuje promene u okviru kolekcije '
    lvPokemons'.
    // Time je obezbedjeno da na grafickom interfejsu korisnik uvek ima pogled (eng.
    view) na
    // azurnu verziju stanja kolekcije koja se posmatra.
    lvPokemons.setItems(pokemons)
  }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="
    -Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/8
    " xmlns:fx="http://javafx.com/fxml/1" fx:controller="example02.PokemonController"
    >
  <columnConstraints>
    <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
    <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  </rowConstraints>
   <children>
      <ListView fx:id="lvPokemons" prefHeight="200.0" prefWidth="200.0" GridPane.
    columnIndex="1" />
      <VBox prefHeight="200.0" prefWidth="100.0" spacing="10.0" GridPane.halignment="
```



Slika 7.4: *Sortiranje rastuće*



Slika 7.5: *Sortiranje opadajuće*

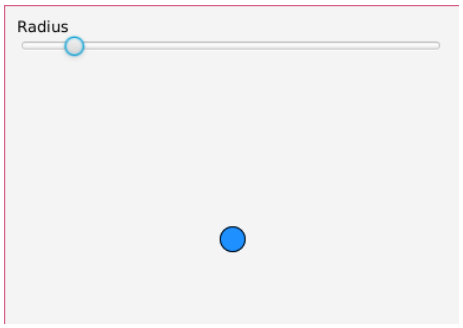
```
    CENTER" GridPane.valignment="CENTER">
21          <children>
               <RadioButton fx:id="rbAsc" mnemonicParsing="false" text="Ascending" />
23             <RadioButton fx:id="rbDesc" mnemonicParsing="false" text="Descending" />
               <Button fx:id="btSort" mnemonicParsing="false" onAction="#handleSort"
    text="Sort" />
25          </children>
         </VBox>
27    </children>
    <padding>
29       <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
    </padding>
31 </GridPane>
```

Zadatak 7.3 Napisati program koji omogućava korisniku da pomeranjem kontroler `Slider` u realnom vremenu menja poluprečnik kruga. Koristeći alat `SceneBuilder` napraviti grafički korisnički interfejs kao na slikama ?? i ??. Prikazani su kodovi redom za datoteke:

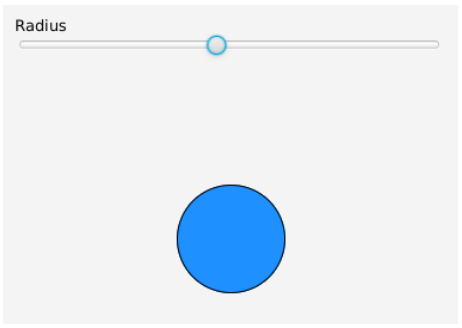
- `CircleApp.scala`
- `MainController.scala`
- `mainLayout.fxml`

```
1  package example03

3  import java.io.IOException
   import javafx.fxml.FXMLLoader
5  import javafx.scene.{Parent, Scene}

7  import scalafx.Includes._
   import scalafx.application.JFXApp
9  import scalafx.application.JFXApp.PrimaryStage

11 object CircleApp extends JFXApp {
     val resource = getClass.getResource("mainLayout.fxml")
13   if (resource == null) {
       throw new IOException("Neuspelo ucitavanje resursa mainLayout.fxml")
15   }

17   val root: Parent = FXMLLoader.load(resource)

19   stage = new PrimaryStage {
       title = "Circle"
21     scene = new Scene(root)
     }
23 }
```

```
   package example03
2
   import java.net.URL
4  import java.util.ResourceBundle
   import javafx.beans.value.{ChangeListener, ObservableValue}
6  import javafx.fxml.{FXML, Initializable}
   import javafx.scene.control.Slider
8  import javafx.scene.shape.Circle

10 class MainController extends Initializable {
```



Slika 7.6: *Promena poluprečnika kruga 1*



Slika 7.7: *Promena poluprečnika kruga 2*

```
  @FXML private var slider: Slider = _
  @FXML private var circle: Circle = _

  override def initialize(location: URL, resources: ResourceBundle): Unit = {
    slider.valueProperty().addListener(new ChangeListener[Number] {
      override def changed(observable: ObservableValue[_ <: Number], oldValue: Number
    , newValue: Number): Unit = {
        setCircleRadius(newValue.intValue())
      }
    })
  }

  def setCircleRadius(r: Int) {
    circle.setRadius(r)
  }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.shape.*?>
<?import javafx.scene.text.*?>
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="
    -Infinity" prefHeight="400.0" prefWidth="499.0" xmlns="http://javafx.com/javafx/8
    " xmlns:fx="http://javafx.com/fxml/1" fx:controller="example03.MainController">
  <columnConstraints>
    <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
  </columnConstraints>
  <rowConstraints>
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
  </rowConstraints>
   <children>
      <VBox prefHeight="200.0" prefWidth="100.0" GridPane.halignment="CENTER"
    GridPane.valignment="CENTER">
         <children>
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Radius" />
            <Slider fx:id="slider" />
         </children>
      </VBox>
      <Circle fx:id="circle" fill="DODGERBLUE" radius="100.0" stroke="BLACK"
    strokeType="INSIDE" GridPane.halignment="CENTER" GridPane.rowIndex="1" GridPane.
    valignment="CENTER" />
   </children>
   <padding>
      <Insets bottom="10.0" left="10.0" right="10.0" top="10.0" />
   </padding>
</GridPane>
```

7.1.3 Zadaci za samostalni rad sa rešenjima

Zadatak 7.4 Napraviti aplikaciju koja prikazuje matricu sa tablicom množenja. Elementi matrice su dugmići na kojima je ispisana vrsta i kolona matrice kojoj pripadaju. Klikom na odgovarajući dugmić na standardni izlaz se prikazuje rezultat množenja. Omogućiti da veličina tablice množenja bude promenljiva korišćenjem slajdera. Vrednost slajdera definiše dimenziju matrice (max=9) koja se prikazuje u centralnom delu (na primer, ako je vrednost slajdera 5 potrebno je da se prikaže tablica množenja veličine 5x5). Za osnovni kontejner upotrebiti BorderPane na kom se u gornjem regionu dodaje FlownPane sa slajderom čija je inicijalna vrednost 0 i tekstom "Veličina tablice množenja". Matricu dugmića dodati u GridPane koji se postavlja u centralnom regionu BorderPane kontejnera.

[Rešenje 7.5]

Zadatak 7.5 Napraviti aplikaciju koja organizuje grupe za zajednički rad na seminarskom zadatku. Program pri pokretanju daje svoj predlog za dve grupe u obliku listi u kojima su zapisana imena studenata (u svakoj po 5). Omogućiti korisniku da napravi drugačije grupe prebacivanjem studenata iz jedne u drugu listu. Za osnovni kontejner upotrebiti HBox koji sadrži dve liste. Između listi treba da stoje dugmići označeni sa »>"i «<"(za njihovu organizaciju koristiti VBox). Odabirom elementa iz leve liste i klikom na »>", taj element je potrebno prebaciti na kraj desne liste (analogno i za dugme «<").

[Rešenje 7.6]

7.2 Rešenja

Rešenje 7.5

```
import scalafx.Includes._
import scalafx.application.JFXApp
import scalafx.application.JFXApp.PrimaryStage
import scalafx.stage.Stage
import scalafx.scene._
import scalafx.scene.input._
import scalafx.geometry._
import scalafx.scene.control._
import scalafx.scene.layout._
import scalafx.scene.text._
import scalafx.scene.shape._
import scalafx.scene.paint.Color
import javafx.beans.value.{ChangeListener, ObservableValue}


object SlajderMatrica extends JFXApp {

  /**
   * Pravimo komponentu glavnog okvira
   * */
  val osnova = new BorderPane {}

  /**
   * Pravimo komponentu okvira za centralni region
   * */
  val top = new FlowPane {
    alignment = Pos.Center
  }

  /**
   * Pravimo tekstualnu komponentu
   * i dodajemo je u gornji region
   * */
  val lab= new Text {
    text = "Velicina tablice mnozenja"
  }
  top.children.add(lab)
```

```
38
   /**
40  *  Pravimo komponentu klizaca
    *  i dodajemo je u gornji region
42  * */
   val slajder = new Slider {
44   min = 0
     max = 9
46   value = 0
     showTickLabels = true
48   blockIncrement = 1
   }
50 top.children.add(slajder)

52 /**
    * Pravimo komponentu okvira
54  * za srednji region
    * i dodajemo je glavnom okviru
56  * */
   val grid = new GridPane {
58   alignment = Pos.Center
   }
60 osnova.setCenter(grid)

62 /**
    * Definisemo funkciju koja ce da se izvrsava
64  * kada se desi promena vrednosti klizaca
    * */
66 slajder.value.addListener(new ChangeListener[Any] {
     def changed(observable: ObservableValue[_], oldValue: Any, newValue: Any) {
68     /**
        * Cistimo centralni okvir
70      * */
       grid.children.clear()
72
       /**
74      * Pravimo dugmice za sve vrednosti matrice mnozenja
        * */
76     for(i <- 1 to newValue.toString.toDouble.toInt)
         for(j <- 1 to newValue.toString.toDouble.toInt) {
78         val dugme = new Button {
             text = "("+i+","+j+")"
80         }

82        /**
            * Definisemo funkciju koja ce da se izvrsava
84          * kada se desi klik misa na neko dugme,
            * ispisujemo pomnozenu vrednost,
86          * i dodajemo dugme u centralni okvir
            * na poziciju (i,j)
88          * */
           dugme.handleEvent(MouseEvent.MousePressed)({
90             dogadjaj: MouseEvent => {
                 println(i+" * "+j+" = "+i*j)
92           }
           })
94         grid.add(dugme, i, j)
       }
96   }
    }
98 )

100 osnova.setTop(top)

102 stage = new PrimaryStage {
     scene = new Scene(440, 270) {
104    content = osnova
     }
106 }
}
```

Rešenje 7.6

```
import scalafx.Includes._
import scalafx.application.JFXApp
import scalafx.application.JFXApp.PrimaryStage
import scalafx.collections.ObservableBuffer
import scalafx.stage.Stage
import scalafx.scene._
import scalafx.scene.input._
import scalafx.geometry._
import scalafx.scene.control._
import scalafx.scene.layout._
import scalafx.scene.text._
import scalafx.scene.shape._
import scalafx.scene.paint.Color
import scalafx.event._
import scala.collection.mutable.ListBuffer

object DveListe extends JFXApp {
  /**
   * Pravimo komponentu glavnog okvira
   * */
  val osnova = new HBox {
    alignment = Pos.Center
    padding = Insets(10,10,10,10)
    prefWidth = 400
    prefHeight = 200
  }

  /**
   * Pravimo prvu listu imena kao i komponentu koja je cuva
   * i dodajemo je u glavni okvir
   * */
  var imena1 = ListBuffer("Marko", "Jana", "Petar", "Milica", "Zeljko")

  val lista1 = new ListView[String] {
      items = ObservableBuffer(imena1)
      orientation = Orientation.Vertical
  }
  osnova.children.add(lista1)

  /**
   * Pravimo komponentu dugmeta za prebacivanje
   * iz leve u desnu listu
   * */
  val dugme1 = new Button {
      text = ">>"
      minWidth = 100
  }
   /**
   * Definisemo funkciju koja ce da se izvrsava
   * kada se desi klik misa na dugme,
   * dohvatamo selektovani element liste
   * i prebacujemo ga u drugu listu
   * */
  dugme1.handleEvent(MouseEvent.MousePressed)({
      dogadjaj : MouseEvent => {
        val idx = lista1.getSelectionModel.getSelectedIndex
        if(idx >= 0){
          val ime = imena1(idx)
          imena1 = imena1 -= ime
          imena2 = imena2 += ime
          lista2.items.setValue(ObservableBuffer(imena2))
          lista1.items.setValue(ObservableBuffer(imena1))
        }
      }
  })

  /**
   * Pravimo drugu listu imena kao i komponentu koja je cuva
   * i dodajemo je u glavni okvir
   * */
  var imena2 = ListBuffer("Ljubica", "Andrea", "Milan", "Darko", "Djordje")

  val lista2 = new ListView[String] {
```

```
        items = ObservableBuffer(imena2)
75      orientation = Orientation.Vertical
    }
77
    /**
79   * Pravimo dugme za prebacivanje
     * iz desne u levu listu
81   * */
    val dugme2 = new Button {
83      text = "<<"
        minWidth = 100
85  }
     /**
87   * Definisemo funkciju koja ce da se izvrsava
     * kada se desi klik misa na dugme,
89   * dohvatamo selektovani element liste
     * i prebacujemo ga u drugu listu
91   * */
    dugme2.handleEvent(MouseEvent.MousePressed)({
93      dogadjaj : MouseEvent => {
          val idx = lista2.getSelectionModel.getSelectedIndex
95        if(idx >= 0){
            val ime : String = imena2(idx)
97          imena2 = imena2 -= ime
            imena1 = imena1 += ime
99          lista2.items.setValue(ObservableBuffer(imena2))
            lista1.items.setValue(ObservableBuffer(imena1))
101       }
        }
103 })

105 /**
     * Pravimo komponentu okvira za srednji region
107  * */
    val sredina = new VBox {
109   alignment = Pos.Center
      prefWidth = 150
111   prefHeight = 150
    }
113
    /**
115  * Dodajemo decu komponente u predvidjene okvire
     * */
117 sredina.children.add(dugme1)
    sredina.children.add(dugme2)
119 osnova.children.add(sredina)
    osnova.children.add(lista2)
121
    stage = new PrimaryStage {
123   scene = new Scene(400, 200) {
        content = osnova
125   }
    }
127 }
```

8

Logičko programiranje

8.1 Jezik Prolog

8.1.1 O jeziku Prolog

Prolog (eng. *PROgramming in LOGic*) je deklarativan programski jezik namenjen rešavanju zadataka simboličke prirode. Prolog se temelji na teorijskom modelu logike prvog reda. Početkom 1970-ih godina Alain Kolmerauer (eng. *Alain Colmeraуer*) i Filipe Rousel (eng. Philippe Roussel) na Univerzitetu u Marselju (eng. University of Aix-Marseille), zajedno sa Robertom Kovalskim (eng. *Robert Kowalski*) sa Odeljka Veštačke Inteligencije (eng. *Department of Artifical Intelligence*) na Univerzitetu u Edinburgu (eng. *University of Edinburgh*), razvili su osnovni dizajn jezika Prolog.

8.1.2 Instalacija BProlog-a

U okviru kursa će biti korišćena distribucija Prologa pod nazivom `BProlog`. `BProlog` se može preuzeti sa zvanične Veb strane http://www.picat-lang.org/bprolog/.

Potrebno je preuzeti adekvatnu verziju za Vaš sistem i otpakovati je. U dobijenom direktorijumu će postojati izvršiva datoteka `bp` kojom se može pokrenuti `BProlog` interpreter. Preporučeno je dodati `bp` u `PATH` kako bi `BProlog` bio dostupan iz komandne linije.

Na primer, pretpostavimo da imamo 64bitni Linux. Potrebno je preuzeti datoteku `bp81_linux64.tar.gz` i smestiti je u direktorijum po izboru, na primer `/home/korisnik/Downloads`. Potom treba izvršiti sledeće naredbe:

```
cd ~/Downloads
tar -xvf bp81_linux64.tar.gz
sudo mv BProlog /opt
sudo ln -s /opt/BProlog/bp /usr/bin/bprolog
```

Nakon toga, `BProlog` interpreter se iz konzole može pokrenuti komandom `bprolog`.

8.2 Uvod

8.2.1 Uvodni primeri

Zadatak 8.1 U bazu znanja uneti informacije o životinjama i njihovim odnosima po pitanju veličine. Napisati pravilo koje omogućava da se proveri koja je od dve životinje veća, kao i da generiše sve životinje za koje je neka životinja veća.

```
% ovo je jednolinijski komentar

/*
ovo je viselinijski
komentar
*/

/*
```

```
Programi se cuvaju sa ekstenzijom .pro ili .pl (na sistemima gde ekstenzije imaju
    globalno znacenje, kao sto je MS-Windows da ne bi bilo mesanja sa Perl programima
     treba uvek koristiti .pro).
Interpreter se pokrece komandom bp. Naredbe:
help -- pomoc
compile('ime_programa') -- prevodi program i pravi izvrsni fajl ukoliko nema gresaka
load('ime_izvrsnog_fajla') -- uvozi izvrsni fajl
cl('ime_programa') -- compile + load
halt ili control+D -- za izlazak iz interpretera

Termovi: konstante, promenljive ili kompozitni termovi.
--- Konstante: atomi i brojevi.
------ Atomi: stringovi od najvise 1000 karaktera koji pocinju malim slovom ('abc', '
    a01', 'b_cd', 'l122k', ...).
------ Brojevi: celi i realni.
--- Promenljive: imena pocinju velikim slovom ili podvlakom (_). Specijalna, anonimna
     promenljiva: '_'.
--- Kompozitni (slozeni) termovi ili strukture: oblika f(t1, ..., tn) gde je f neka
    funkcija arnosti n ( 0<n<32768), a t1, ..., tn termovi.

Program:  sekvenca Hornovih klauza. Postoje tri tipa Hornovih klauza: cinjenice,
    pravila i upiti.
--- Cinjenice: atomicna formula oblika p(t1, ..., tn) gde je p predikat arnosti n, a
    t1, ..., tn termi. One opisuju svojstva i relacije izmedju objekata. Primer:
  zivotinja(slon).
  veci(zebra,vuk).
--- Pravila: imaju sledecu formu
  H :- B1, ..., Bn. (n>0)
H, B1, ..., Bn su atomicne formule. H se zove GLAVA pravila, dok je sve sa desne
    strane :- TELO pravila. Citamo ih kao implikaciju sa desna na levo: vazi H, ako
    vaze B1, ..., Bn ("," u pravilu zamenjuje logicko "i").
--- Cinjenice i pravila cine BAZU ZNANJA.
--- Upiti: konstrukcije kojima korisnik komunicira sa bazom znanja. Za ovo je
    neophodan interpetator kome se postavlja upit. Primer:
  ?- veci(slon, zec).
  true.
  ?- zivotinja(veverica).
  false.
*/

/* cinjenice, svojstva */
zivotinja(slon).
zivotinja(vuk).
zivotinja(zec).
zivotinja(zebra).

/* cinjenice, odnosi */
veci(slon,vuk).
veci(vuk,zec).
veci(slon,zebra).
veci(zebra,vuk).
veci(slon,zec).

/*
upiti sa promenljivama:
-- daje jedno resenje, ako zelimo da prikaze jos resenja kucamo ; nakon prikazanog ?
    a za prekid control+C

?- veci(slon, X).
X = vuk ? ^C

| ?- veci(slon, X).
X = vuk ?;
X = zebra ?;
X = zec
yes

| ?- veci(X, Y).
X = slon
Y = vuk ?;
X = vuk
Y = zec ?;
X = slon
```

```
Y = zebra ?;
X = zebra
Y = vuk ?;
X = slon
Y = zec
yes
*/

/*
pravilo : za neko X i Y vazi je_veci(X,Y) ako postoji Z tako da vazi veci(X,Z) i veci
    (Z,Y)
*/
je_veci(X,Y):-veci(X,Z),veci(Z,Y).
/*
| ?- je_veci(X,Y)
X = slon
Y = zec ?;
X = slon
Y = vuk ?;
X = zebra
Y = zec ?;
no
*/
```

Zadatak 8.2 Unifikacija. Jednakost.

```
/*

Provera tipa:

atom(X) - da li je term X atom
atomic(X) - da li je term X atom ili broj
number(X) -  da li je term X broj
float(X) ili real(X) - da li je term X realan broj
integer(X) - da li je term X ceo broj
var(X) -  da li je term X slobodna promenljiva
nonvar(X) - da li term X nije promenljiva

Primeri:
| ?- atom('abc')
yes
| ?- atomic(5)
yes

Unifikacija:

=   unifikabilni
\=  nisu unifikabilni
==  identicno jednaki termovi
\== nisu identicno jednaki termovi

*/

uni(X, Y):- X = Y.

/*
| ?- uni(4,5)
no
| ?- uni(4,X)
X = 4
yes
*/

jed(X, Y):- X == Y.

/*
| ?- jed(4,X)
no
| ?- jed(4,5)
no
| ?- jed(4,4)
yes
*/
```

Zadatak 8.3 Aritmetički operatori. Operatori *is* i *cut*.

```
/*
is   aritmeticko izracunavanje
=:=  aritmeticki jednaki
=\=  aritmeticki nisu jednaki
<, =<, >, >=
+, -, *, /, // (celobrojno deljenje), div, mod, ** (stepenovanje)
*/

/*
Ako je X promenljiva, tada se njoj dodeljuje vrednost koju ima term Y (mora biti
    poznata vrednost), a ukoliko X nije promenljiva, X is Y se svodi na X =:= Y
*/
op1(X, Y):- X is Y.
/*
Termovima X i Y moraju biti poznate vrednosti, inace ce prijaviti gresku.
*/
op2(X, Y):- X =:= Y.

/*
| ?- op1(3,4)
no
| ?- op1(4,4)
yes
| ?- op1(X,4)
X = 4
yes
| ?- op1(4,X)
*** error
| ?- op2(4,4)
yes
| ?- op2(4,2)
no
| ?- op2(4,X)
*** error
| ?- op2(X,4)
*** error
*/

/*
apsolutna vrednost, prvi argument je broj za koji trazimo apsolutnu vrednost, a drugi
     promenljiva gde se smesta rezultat

losa implementacija, za pozitivne brojeve oba pravila prolaze

| ?- abs1(1,X)
X = 1 ?;
X = -1
yes
| ?- abs1(-1,X)
X = 1
yes

*/
abs1(X, X):- X >= 0.
abs1(X, Y):- Y is -X.

/*
dobre implementacije abs2 i abs3

| ?- abs2(1,X)
X = 1 ?
yes
| ?- abs2(-1,X)
X = 1
yes
*/
abs2(X, X):- X >= 0.
abs2(X, Y):- X < 0, Y is -X.
```

```
/*
koriscenje operatora "cut" koji se oznacava sa "!" uklanjamo alternativne klauze,
    tako da se u slucaju pozitivnih nece primeniti drugi predikat cim uspe prvi

| ?- abs3(1,X)
X = 1
yes
| ?- abs3(-1,X)
X = 1
yes

*/
abs3(X, X):- X >= 0, !.
abs3(X, Y):- Y is -X.
```

Zadatak 8.4 Rekurzivni predikat, primer porodičnog stabla.

```
% porodicno stablo

% svojstva
musko(mihajlo).
musko(stevan).
musko(petar).
musko(mladen).
musko(rajko).
zensko(milena).
zensko(milica).
zensko(jelena).
zensko(senka).
zensko(mina).
zensko(maja).

% odnosi
roditelj(mihajlo,milica).
roditelj(mihajlo,senka).
roditelj(milena,rajko).
roditelj(maja,petar).
roditelj(maja,mina).
roditelj(stevan,mladen).
roditelj(stevan,jelena).
roditelj(milica,mladen).
roditelj(milica,jelena).

% pravila
majka(X,Y):- roditelj(X,Y), zensko(X).
otac(X,Y):- roditelj(X,Y), musko(X).
brat(X,Y):- musko(X), majka(Z,X), majka(Z,Y), X\==Y.
sestra(X,Y):- zensko(X), majka(Z,X), majka(Z,Y), X\==Y.
ujak(X,Y):- brat(X,Z), majka(Z,Y).
tetka(X,Y):- sestra(X,Z), majka(Z,Y).

% rekurzivno pravilo
% roditelj je predak
predak(X,Y):- roditelj(X,Y).
% roditelj pretka je takodje predak
predak(X,Y):- roditelj(X,Z), predak(Z,Y).
```

Zadatak 8.5 Napisati Prolog predikate:

- `prestupna` koji određuje da li je godina prestupna
- `brdana` koji određuje koliko dana ima prosleđeni mesec

```
% godina je prestupna ako je deljiva sa 4 i nije deljiva sa 100 ili je deljiva sa 400
prestupna(X):- X mod 4 =:= 0, X mod 100 =\= 0.
prestupna(X):- X mod 400 =:= 0.

/*
anonimna promenljiva _ se koristi da oznaci da nam vrednost koja se prosledi za
    godinu nije bitna, moze biti bilo sta, ali tu vrednost ne koristimo

```

```
| ?- brdana(januar, _, X)
X = 31
yes
| ?- brdana(januar, 2017, X)
X = 31
yes
*/
brdana(januar, _, 31).
brdana(februar, X, 28):- not(prestupna(X)).
brdana(februar, X, 29):- prestupna(X).
brdana(mart,_,31).
brdana(april,_,30).
brdana(maj,_,31).
brdana(jun,_,30).
brdana(jul,_,31).
brdana(avgust,_,31).
brdana(septembar,_,30).
brdana(oktobar,_,31).
brdana(novembar,_,30).
brdana(decembar,_,31).
```

8.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 8.6 Napisati sledeće predikate:

a) `maksimum(A, B, M)` - određuje maksimum za dva broja `A` i `B`

b) `suma(N, S)` - za dati prirodan broj `N` računa sumu prvih `N` brojeva

c) `sumaParnih(N, S)` - za dati paran prirodan broj `N` računa sumu parnih brojeva od 2 do `N`

d) `proizvod(N, P)` - za dati prirodan broj `N` računa proizvod prvih `N` prirodnih brojeva

e) `proizvodNeparnih(N, P)` - za dati neparan prirodan broj `N` računa proizvod neparnih brojeva od 1 do `N`

f) `cifre(N)` - ispisuje cifre prirodnog broja `N` rečima

[Rešenje 8.6]

8.2.3 Zadaci za vežbu

Zadatak 8.7 Napisati sledeće predikate:

a) `sumaCifara(N, SC)` - određuje sumu cifara prirodnog broja `N`

b) `brojCifara(N, BC)` - određuje broj cifara prirodnog broja `N`

c) `maxCifra(N, MC)` - određuje maksimalnu cifru prirodnog broja `N`

d) `sumaKvadrata(N, SK)` - računa sumu kvadrata prvih `N` prirodnih brojeva

e) `fakt(N, F)` - računa faktorijel prirodnog broja `N`

g) `sumaDel(X, D)` - računa sumu pravih delilaca broja `X`

Zadatak 8.8 Ako su date činjenice oblika:

- `ucenik( SifraUcenika, ImeUcenika, Odeljenje)`
- `ocene( SifraUcenika, SifraPredmeta, Ocena)`
- `predmet( SifraPredmeta, NazivPredmeta, BrojCasova)`

Napisati sledeće predikate:

a) `bar2PeticeSifra(S)` - određuje šifru `S` učenika koji ima bar dve petice iz različitih predmeta

b) `bar2PeticeIme(X)` - određuje ime `X` učenika koji ima bar dve petice iz različitih predmeta

c) `odeljenjePetice(X,Y)` - određuje odeljenje `X` u kome postoje bar dve petice iz predmeta sa šifrom `Y`

Zadatak 8.9 Ako su date činjenice oblika:

- `film(NazivFilma, ZanrFilma, ImeReditelja, SifraGlumca)`
- `glumac(SifraGlumca, ImeGlumca, GodRodj, MestoRodj)`

Napisati sledeće predikate:

a) `filmskiUmetnik(X)` - `X` je filmski umetnik ako je reditelj nekog filma i igra u nekom filmu

b) `glumacBarDva(X)` - određuje ime glumca `X` koji igra u bar dva različita filma

c) `opstiGlumac(X)` - određuje ime glumca `X` koji igra u bar dva filma različitog žanra

d) `zanrovskiGlumac(X,Y)` - određuje ime glumca `X` koji igra u filmu žanra `Y`

8.3 Liste

8.3.1 Uvodni primeri

Zadatak 8.10 Osnovni pojmovi i predikati za rad sa listama.

```
/*
Lista - niz uredjenih elemenata, tj. termova.
Lista moze biti:
  [] - prazna
  .(G,R) - struktura, gde je G ma koji term i naziva se glava liste, a R lista i
    naziva se rep liste

Primeri:
  [] - prazna
  .(a, []) - jednoclana lista, gde je a bilo koji term
  .(a, .(b, [])) - dvoclana lista, gde su a i b termi ...

Zapis pomocu zagrada (prvi element predstavlja glavu, a ostali cine listu koja je rep
    ):
  [a,b,c] <=> .(a, .(b, [c]))

Zapis liste u kom su jasno razdvojeni glava i rep (pogodan za unifikaciju): [G|R].

Primeri unifikacije listi:
[X, Y, Z] [jabuka, kruska, banana]  -----> X = jabuka, Y = kruska, Z = banana
[racunar] [X|Y]                     -----> X = racunar, Y = []
[maja, ana, jovana] [X, Y|Z]          -----> X = maja, Y = ana, Z = [jovana]

*/

% predikat proverava da li element pripada listi (ako joj pripada jednak je glavi ili
    nekom elementu iz repa liste)
sadrzi(X, [X|_]):- !.
sadrzi(X, [G|R]):- G \== X, sadrzi(X, R).

% drugi nacin, predikat kao disjunkcija:
% sadrzi(X, [G|R]):- G == X; sadrzi(X, R).

% predikat koji racuna duzinu liste (prazna je duzine nula, nepraznu dekomponujemo na
    glavu i rep, pa je duzina liste = 1 + duzina repa)
duzina([], 0).
duzina([G|R], L):- duzina(R,L1), L is L1+1.

% predikat racuna sumu elemenata liste brojeva
```

```
suma([], 0).
suma([G|R], S):- number(G), suma(R, S1), S is S1+G.

% predikat racuna aritmeticku sredinu elemenata liste brojeva
arsr([],0).
% ako ne koristimo sablon za nepraznu listu [G|R], moramo proveriti da li je K nula
    jer se sada L moze unifikovati sa []
arsr(L, A):- duzina(L, K), K =\= 0, suma(L, S), A is S/K.

% predikat ucitava listu duzine N ciji elementi mogu biti proizvoljni termovi
% za negativno N ne ucitava listu
ucitaj(N,_) :- N < 0, !.
% za nulu vraca praznu listu
ucitaj(0, []).
% read(X) ucitanu vrednost sa ulaza dodeljuje promenljivoj X
% ako je N > 1, lista ima glavu i rep, ucitamo glavu, pa pozovemo predikat za
    ucitavanje repa
ucitaj(N, [G|R]):- N > 1, write('unesi element '), read(G), nl, M is N-1, ucitaj(M,R)
    .

/*
prilikom unosa vrednosti obavezna je tacka kao oznaka kraja ulaza za read
| ?- ucitaj(3,L)
unesi element | 5.

unesi element | 4.

unesi element | 3.

L = [5,4,3]
yes

| ?- ucitaj(3,L)
unesi element | [1,2,3].

unesi element | 4.

unesi element | [].

L = [[1,2,3],4,[]]
yes

*/
```

8.3.2 Zadaci za samostalni rad sa rešenjima

Zadatak 8.11 Napisati sledeće predikate:

a) `dodajPocetak(X, L, NL)` - dodaje `X` na početak liste `L`

b) `dodajKraj(X, L, NL)` - dodaje `X` na kraj liste `L`

c) `obrisiPrvi(L, NL)` - briše prvi element, tj. glavu liste

d) `obrisiPoslednji(L, NL)` - briše poslednji element liste

e) `obrisi(X, L, NL)` - briše sva pojavljivanja elementa `X` u listi `L`

f) `obrisiPrvo(X, L, NL)` - briše samo prvo pojavljivanje elementa `X` u listi `L`

g) `obrisiK(L, K, NL)` - briše `K`-ti element liste `L`

[Rešenje 8.11]

Zadatak 8.12 Napisati predikat `podeli(L, L1, L2)` koji deli listu `L` na dve liste, listu pozitivnih elemenata `L1` i listu negativnih elemenata `L2`.

[Rešenje 8.12]

Zadatak 8.13 Napisati predikat `dupliraj(L, NL)` koji od date liste `L` formira novu listu `NL` tako što svaki negativan element duplira, tj. dva puta upisuje u novu listu.

[Rešenje 8.13]

Zadatak 8.14 Napisati predikat `zameni(X, Y, L, NL)` koji od date liste `L` formira novu listu `NL` zamenom elemenata `X` i `Y`.

[Rešenje 8.14]

Zadatak 8.15 Napisati predikat `pretvori(L, X)` koji za datu listu cifara `L` formira broj određen tim ciframa.

[Rešenje 8.15]

Zadatak 8.16 Napisati predikat `maxEl(L, X)` koji određuje maksimalni element liste `L`.

[Rešenje 8.16]

Zadatak 8.17 Napisati predikate za sortiranje liste rastuće:

a) `insertionSort(L, SL)` - insertion sort algoritam se zasniva na ubacivanju redom svakog elementa liste na svoje pravo mesto (mesto u sortiranoj listi)

b) `mergeSort(L, SL)` - merge sort algoritam se zasniva na dekompoziciji liste, tj. listu delimo na dva jednaka dela, te delove sortiramo i posle toga ih objedinjujemo

[Rešenje 8.17]

8.3.3 Zadaci za vežbu

Zadatak 8.18 Napisati predikat `parNepar(L, L1, L2)` koji deli listu `L` na dve liste, listu parnih elemenata `L1` i listu neparnih elemenata `L2`.

Zadatak 8.19 Napisati predikat `podeli(L, N, L1, L2)` koji deli listu `L` na dve liste `L1` i `L2`, pri čemu je zadata dužina prve liste `L1`.

Zadatak 8.20 Napisati predikat `ogledalo(L1, L2)` koji proverava da li je lista `L1` jednaka obrnutoj listi liste `L2`.

Zadatak 8.21 Napisati predikat `interval(X, Y, L)` koji kreira listu `L` koja sadži sve cele brojeve iz intervala zadatog sa prva dva argumenta.

Zadatak 8.22 Napisati predikat `skalar(L1, L2, S)` koji određuje skalarni proizvod dva vektora, tj. listi brojeva `L1` i `L2`.

Zadatak 8.23 Napisati predikat `sortirana(L)` koji proverava da li je lista `L` sortirana, bilo opadajuće ili rastuće.

Zadatak 8.24 Napisati predikat `spoji(L1, L2, L)` koji spaja dve rastuće sortirane liste `L1` i `L2` u treću tako da i ona bude sortirana rastuće.

8.4 Razni zadaci

8.4.1 Zadaci sa rešenjima

Zadatak 8.25 Napisati predikate `nzd(N, M, NZD)` i `nzs(N, M, NZS)` koji određuju najveći zajednički delilac i najmanji zajednički sadržalac prirodnih brojeva `N` i `M` redom.

[Rešenje 8.25]

Zadatak 8.26 Ako su date činjenice oblika:

- `stan(Porodica, KvadraturaStana)`
- `clan(Porodica, BrojClanova)`

Napisati predikat `poClanu(Porodica, Prosek)` koji određuje prosečan broj kvadrata stana po članu porodice koja živi u njemu.

[Rešenje 8.26]

Zadatak 8.27 Ako je data baza znanja:

- `automobil(SifraAutomobila, NazivAutomobila)`
- `vlasnik(ImeVlasnika, SifraAutomobila)`
- `brziSifra(SX, SY)` - automobil šifre `SX` je brži od automobila šifre `SY`

Napisati predikate:

a) `brziNaziv(X, Y)` - automobil naziva `X` je brži od automobila naziva `Y`

b) `imaAutomobil(X)` - `X` je vlasnik nekog automobila

c) `imaBrzi(X, Y)` - `X` je vlasnik bržeg automobila od onog čiji je vlasnik `Y`.

[Rešenje 8.27]

Zadatak 8.28 Napisati predikat `savrsen(N)` koji proverava da li je prirodan broj `N` savršen, tj. da li je jednak sumi svojih pravih delilaca. U slučaju da se prosledi neispravan argument, predikat treba da ispiše poruku o grešci i prekine program.

[Rešenje 8.28]

Zadatak 8.29 Napisati predikat `izbaci3(N, X)` koji iz prirodnog broja `N` izbacuje sve cifre manje 3. U slučaju da se prosledi neispravan argument, predikat treba da prekine program.

[Rešenje 8.29]

Zadatak 8.30 Napisati predikate za liste brojeva:

a) `duplikati(L, L1)` - izbacuje duplikate iz liste `L`

b) `unija(L1, L2, L)` - određuje uniju listi `L1` i `L2`

c) `presek(L1, L2, L)` - određuje presek listi `L1` i `L2`

d) `razlika(L1, L2, L)` - određuje razliku listi `L1` i `L2`

[Rešenje 8.30]

Zadatak 8.31 Napisati program koji rešava sledeću zagonetku. Postoji pet kuća, svaka različite boje u kojoj žive ljudi različitih nacionalnosti koji piju različita pića, jedu različita jela i imaju različite kućne ljubimce. Važi sledeće:

- Englez živi u crvenoj kući
- Španac ima psa
- kafa se pije u zelenoj kući
- Ukrajinac pije čaj
- zelena kuća je odmah desno uz belu
- onaj koji jede špagete ima puža
- pica se jede u žutoj kući
- mleko se pije u srednjoj kući

- Norvežanin živi u prvoj kuci s leva
- onaj koji jede piletinu živi pored onoga koji ima lisicu
- pica se jede u kući koja je pored kuće u kojoj je konj
- onaj koji jede brokoli pije sok od narandze
- Japanac jede suši
- Norvežanin živi pored plave kuće

Čija je zebra, a ko pije vodu?

[Rešenje 8.31]

Zadatak 8.32 Napisati program koji rešava sledeću zagonetku. Svakog vikenda, Milan čuva petoro komšijske dece. Deca se zovu Kata, Lazar, Marko, Nevenka i Ognjen, a prezivaju Filipović, Grbović, Hadžić, Ivanović i Janković. Svi imaju različit broj godina od dve do šest. Važi sledeće:

- jedno dete se zove Lazar Janković
- Kata je godinu dana starija od deteta koje se preziva Ivanović koje je godinu dana starije od Nevenke
- dete koje se preziva Filipović je tri godine starije od Marka
- Ognjen je duplo stariji od deteta koje se preziva Hadžić

Kako se ko zove i koliko ima godina?

[Rešenje 8.32]

8.4.2 Zadaci za vežbu

Zadatak 8.33 Napisati predikat `uzastopni(X, Y, Z, L)` koji proverava da li su prva tri argumenta uzastopni elementi u listi `L`.

Zadatak 8.34 Napisati predikat `kompresuj(L, KL)` koji u datoj listi `L` eliminiše uzastopne duplikate.

Zadatak 8.35 Napisati predikat `prefiksi(L, P)` koji određuje sve liste koje su prefiksi date liste `L`.

Zadatak 8.36 Napisati predikat `sufiksi(L, S)` koji određuje sve liste koje su sufiksi date liste `L`.

Zadatak 8.37 Napisati predikat `opadajuce(N, L)` koji za dat prirodan broj `N` formira listu brojeva od `N` do `1`.

Zadatak 8.38 Napisati predikat `form(N, L)` kojim se formira lista od prirodnih brojeva deljivih sa 5 i manjih od datog prirodnog broja `N`.

Zadatak 8.39 Napisati program koji rešava sledeću zagonetku. Četiri žene se zovu Petra, Milica, Lenka i Jovana, a prezivaju Perić, Mikić, Lazić i Jović. One imaju četiri kćerke koje se takodje zovu Petra, Milica, Lenka i Jovana. Važi sledeće:

- nijedna majka nema prezime koje počinje istim slovom kao ime
- nijedna kćerka nema prezime koje počinje istim slovom kao ime
- nijedna kćerka se ne zove kao majka
- majka koja se preziva Perić se zove isto kao Miličina kćerka
- Lenkina kćerka se zove Petra

Odrediti imena majki i kćerki.

Zadatak 8.40 Napisati program koji rešava sledeću zagonetku. Četiri para je došlo na maskenbal:

- Markova zena se maskirala kao macka
- dva para su stigla pre Marka i njegove žene, a jedan muskarac je bio maskiran u medveda
- prvi koji je stigao nije bio Vasa, ali je stigao pre onoga koji je bio maskiran u princa
- žena maskirana u vešticu (nije Bojana) je udata za Peru, koji se maskirao kao Paja patak
- Marija je došla posle Laze, a oboje su stigli pre Bojane
- žena maskirana u Ciganku je stigla pre Ane, pri čemu nijedna od njih nije udata za muškarca maskiranog u Betmena
- žena maskirana u Snežanu je stigla posle Ivane

Odrediti kako je bio obučen koji par.

Zadatak 8.41 Izračunavanje vrednosti aritmetičkog izraza korišćenjem listi možete pogledati ovde:
https://rosettacode.org/wiki/Arithmetic_evaluation#Prolog

Zadatak 8.42 Implementacije raznih problema u Prologu možete pogledati ovde:
https://rosettacode.org/wiki/Category:Prolog

8.5 Rešenja

Rešenje 8.6

```
% maksimum dva broja
% I nacin:
maksimum(A,B,M):- A>=B, M is A.
maksimum(A,B,M):- A<B, M is B.
% II nacin bez trece promenjive:
% maksimum(A,B,A):- A>=B.
% maksimum(A,B,B):- A<B.

% suma prvih N prirodnih brojeva
suma(1,1).
suma(N,S):- N>1, N1 is N-1, suma(N1,S1), S is S1+N.

% suma parnih prirodnih brojeva od 2 do N
% moze se dodati provera N mod 2 =:= 0 u pravilu, ali i bez toga sam prepoznaje za
    neparne da je netacan upit jer rekurzijom dodje do sumaParnih(1,S) sto je netacna
     cinjenica u bazi
sumaParnih(2,2).
sumaParnih(N,S):- N>2, N1 is N-2, sumaParnih(N1,S1), S is S1+N.

% proizvod prvih N prirodnih brojeva
proizvod(1,1).
proizvod(N,P):- N>1, N1 is N-1, proizvod(N1,P1), P is P1*N.

% proizvod neparnih prirodnih brojeva od 1 do N
proizvodNeparnih(1,1).
proizvodNeparnih(N,P):- N>1, N1 is N-2, proizvodNeparnih(N1,P1), P is P1*N.

% ispis cifara unetog prirodnog broja N
cifra(0, nula).
cifra(1, jedan).
cifra(2, dva).
cifra(3, tri).
cifra(4, cetiri).
cifra(5, pet).
```

```
cifra(6, sest).
cifra(7, sedam).
cifra(8, osam).
cifra(9, devet).

% ukoliko nije prirodan broj, cut operatorom sprecavamo poziv poslednjeg predikata
cifre(N):- N < 1, !.

% ukoliko je jednocifren svodi se na poziv predikata cifra
% write(t) gde je t neki term, ispisuje term
% nl (newline) - ispisuje se novi red
% obratiti paznju na upotrebu cut operatora ! - sprecavamo poziv poslednjeg predikata za jednocifrene
cifre(N):- N > 1, N < 10, cifra(N, C), write(C), nl, !.

% ukoliko nije jednocifren, racunamo tekucu cifru koju ispisujemo i ostatak broja za koji se ponovo poziva predikat
cifre(N):- N1 is (N // 10), cifre(N1), N2 is (N mod 10), cifra(N2, C), write(C), nl.
```

Rešenje 8.11

```
% dodaje element na pocetak  liste
dodajPocetak(X, L, [X|L]).

% dodaje element na kraj liste
dodajKraj(X, [], [X]).
dodajKraj(X, [G|R], [G|LR]):- dodajKraj(X, R, LR).

% brise prvi element liste
% za praznu listu ce uvek vratiti no jer ne moze unifikovati sa sablonom [G|R], a mozemo napraviti i sami za taj slucaj da je predikat netacan
% fail je uvek netacan pa ce nam ovako definisan predikat za slucaj prazne liste vratiti no
obrisiPrvi([], _):- fail.
obrisiPrvi([_|R], R).

% brise poslednji element liste
obrisiPoslednji([],_):- fail.
% bitan cut operator jer se jednoclana moze upariti sa sablonom [G|R]
obrisiPoslednji([_],[]):- !.
obrisiPoslednji([G|R], [G|R1]):- obrisiPoslednji(R, R1).

% brise element X iz liste ako postoji (svako pojavljivanje elementa X)
obrisi(_, [], []).
obrisi(X, [X|R], R1):- obrisi(X, R, R1), !.
obrisi(X, [G|R], [G|R1]):- G \== X, obrisi(X, R, R1).

% brise element X iz liste ako postoji (samo prvo pojavljivanje elementa X)
obrisiPrvo(X, [], []).
obrisiPrvo(X, [X|R], R):- !.
obrisiPrvo(X, [G|R], [G|R1]):- G \== X, obrisiPrvo(X, R, R1).

% brise K-ti element liste, brojimo od 1, ako je K vece od duzine liste, treci argument je jednak prvom
obrisiK([], K, []):- K > 0.
obrisiK([G|R], 1, R):- !.
obrisiK([G|R], K, [G|R1]):- K>1, K1 is K-1, obrisiK(R, K1, R1).

% druga varijanta predikata: brise K-ti element liste, broji od 1, ali ukoliko zadata lista nema K-ti element, predikat vraca no kao odgovor i nema unifikacije za treci argument
% obrisiK([], K, []):- fail.
% obrisiK([G|R], 1, R):- !.
% obrisiK([G|R], K, [G|R1]):- K>1, K1 is K-1, obrisiK(R, K1, R1).
```

Rešenje 8.12

```
% podeli1 - deli listu na dve liste - listu pozitivnih i listu negativnih elemenata
% L1 - lista pozitivnih, L2 - lista negativnih
podeli([], [], []).
```

```
podeli([G|R], [G|R1], L2):- G >= 0, podeli(R, R1, L2), !.
podeli([G|R], L1, [G|R2]):- G < 0, podeli(R, L1, R2).
```

Rešenje 8.13

```
% svaki negativan element dodajemo dva puta u novu listu, a pozitivne samo jednom
dupliraj([], []).
dupliraj([G|R], [G,G|R1]):- G<0, dupliraj(R, R1), !.
dupliraj([G|R], [G|R1]):- G>=0, dupliraj(R, R1).
```

Rešenje 8.14

```
% menja elemente X i Y u listi
zameni(X, Y, [], []).
zameni(X, Y, [X|R], [Y|R1]):- zameni(X, Y, R, R1), !.
zameni(X, Y, [Y|R], [X|R1]):- zameni(X, Y, R, R1), !.
zameni(X, Y, [G|R], [G|R1]):- G \== X, G \== Y, zameni(X, Y, R, R1).
```

Rešenje 8.15

```
% potreban nam je dodatni predikat za izdvajanje poslednjeg elementa liste
izdvojPoslednji([G], G, []):- !.
izdvojPoslednji([G|R], X, [G|R1]):- izdvojPoslednji(R, X, R1).

% formira broj od date liste cifara
pretvori([], 0):- !.
pretvori(L, X):- izdvojPoslednji(L, Poslednji, Ostatak),
    pretvori(Ostatak, Y),
    X is Poslednji + 10 * Y.

/*
| ?- pretvori([7,0,7], X)
X = 707
yes
| ?- pretvori([0,2,3], X)
X = 23
yes
| ?- pretvori([], X)
X = 0
yes

*/
```

Rešenje 8.16

```
% maksimalni element liste
maxEl([X], M):- M is X, !.
% pozivamo za rep (idemo u dubinu), pa poredimo maksimalni element repa i glavu liste
maxEl([G|R], X):- maxEl(R, Y), G < Y, X is Y, !.
maxEl([G|R], X):- maxEl(R, Y), G >= Y, X is G.
```

Rešenje 8.17

```
/*

Insertion sort algoritam se zasniva na ubacivanju redom svakog elementa liste na
    svoje pravo
mesto.

IH (Induktivna hipoteza) - umemo da sortiramo listu od n-1 elementa.
IK (Induktivni korak) - n-ti element ubacujemo na odgovarajucu lokaciju.

*/
insertionSort([], []).
insertionSort([G|R], SL):- insertionSort(R, S1), ubaciS(G, S1, SL).
```

```
12
   % ubacuje sortirano element X u listu
14 ubaciS(X, [], [X]).

16 % ubaciS(X, [G|R], [X, G|R]):- X=<G.
   % ubaciS(X, [G|R], [G|SL]):- X>G, ubaciS(X, R, SL).
18 /*
    alternativno sa cut operatorom ne moramo
20  da pisemo X>G u drugom predikatu, jer
    kada stavimo cut operator kod prvog, cim
22  on uspe drugi predikat se nece ni pokusavati
   */
24 ubaciS(X, [G|R], [X, G|R]):- X=<G, !.
   ubaciS(X, [G|R], [G|SL]):- ubaciS(X, R, SL).
26
   /*
28
   Merge sort algoritam se zasniva na dekompoziciji. Naime, ulaznu listu delimo na dva
       jednaka
30 dela, te delove sortiramo i posle toga ih objedinjujemo.

32 IH - umemo da sortiramo liste velicine n/2.
   IK - u linearnom vremenu objedinjujemo dve sortirane liste od po n/2 elemenata.
34
   */
36 mergeSort([], []).
   mergeSort([X], [X]).
38 mergeSort(N, SL):- podeli(N, L, R),
        mergeSort(L, L1),
40      mergeSort(R, R1),
        objedini(L1, R1, SL).
42
   % delimo tako sto prvi element stavljamo u prvu particiju
44 % drugi u drugu pri cemu su particije dobijene
   % vec rekurzivno podelom repa
46 podeli([], [], []).
   podeli([X], [X], []):- !.
48 podeli([G1, G2|R], [G1|R1], [G2|R2]):- podeli(R, R1, R2).

50 /*
   Primer: ako objedinjujemo [1,3,5,7] i [5,6,7,8,9] znamo da 1 prethodi svima pa mozemo
        rekurzivno da objedinimo [3,5,7] sa [5,6,7,8,9] i dodamo 1 na pocetak tako
       objedinjene liste
52 */
   objedini(L, [], L).
54 objedini([], L, L).
   objedini([G1|R1], [G2|R2], [G1|R]):- G1<G2, objedini(R1, [G2|R2], R), !.
56 objedini([G1|R1], [G2|R2], [G2|R]):- G1>=G2, objedini([G1|R1], R2, R).
```

Rešenje 8.25

```
1  /*
   za odredjivanje nzd i nzs dva prirodna broja koristicemo Euklidov algoritam
3
   bitno je obezbediti da se pogram ispravno ponasa - kada nadje jedno resenje treba
       obezbediti prekid koriscenjem cut operatora u 1. klauzi nzdPom, jer bi u
       suprotnom program zasao u skup negativnih brojeva (pa bi nastao beskonacan ciklus
       ) ili pokusaj deljenja sa nulom (sto bi prijavilo gresku)
5  */
   nzd(A,B,NZD):- A>=B, B>0, nzdPom(A,B,NZD),!.
7  nzd(A,B,NZD):- A<B, A>0, nzdPom(B,A,NZD).

9  nzdPom(N, 0, N):- !.
   nzdPom(N, M, NZD):- P is N mod M, nzdPom(M, P, NZD).
11
   % nzs dobijamo mnozenjem oba prirodna broja i deljenjem sa nzd
13 nzs(N, M, NZS):- nzd(N, M, P), NZS is N*M//P.

15 /*
   | ?- nzd(15,9,NZD)
17 NZD = 3
   yes
```

```
| ?- nzd(12,8,NZD)
NZD = 4
yes
| ?- nzs(13,17,NZS)
NZS = 221
yes
*/
```

Rešenje 8.26

```
% cinjenice
stan(petrovic, 76).
stan(ciric, 93).
stan(aleksic, 55).
stan(lisic, 123).
stan(peric, 67).

clan(ciric, 3).
clan(peric, 5).
clan(aleksic, 2).
clan(lisic, 3).
clan(petrovic, 4).

% koristimo operator za realno deljenje / (// je za celobrojno, ne mesati ova dva
    operatora) da bi se dobio tacan rezultat
poClanu(Porodica, Prosek):- stan(Porodica, X), clan(Porodica, Y), Prosek is X/Y.
```

Rešenje 8.27

```
% baza znanja
automobil(a1, audi).
automobil(h1, honda).
automobil(m1, mercedes).
automobil(m2, mercedes).
automobil(c1, citroen).
automobil(c2, citroen).

vlasnik(milan, h1).
vlasnik(maja, m1).
vlasnik(nemanja, m2).
vlasnik(aleksandar, a1).
vlasnik(andjela, c1).
vlasnik(petar, c2).

brziSifra(a1, c1).
brziSifra(m1, c1).
brziSifra(m2, h1).
brziSifra(a1, c2).

brziNaziv(X, Y):- automobil(SX, X), automobil(SY, Y), brziSifra(SX, SY).

imaAutomobil(X):- vlasnik(X, _).

imaBrzi(X, Y):- vlasnik(X, S1), vlasnik(Y, S2), brziSifra(S1, S2).
```

Rešenje 8.28

```
% fail ce prouzrokovati da 2. klauza uvek bude netacna jer zelimo da se poziv
    predikata za neispravan argument vidi kao netacna cinjenica u bazi
provera(N):- N > 0.
provera(N):- N =< 0, write('Broj nije prirodan'), nl, fail.

% predikat sumaPom za treci argument ima kandidata delioca broja N koji postepeno
    uvecavamo (pocinjemo od jedinice koja deli svaki broj)
sumaDelilaca(N, S):- sumaPom(N, S, 1).
% dovoljno je ici do N/2, suma je inicijalno 0, tako da na primer za N==1 vraca 0
sumaPom(N, 0, I):- I>N//2.
% ako I deli, dodajemo ga na tekucu sumu i pozivamo za sledeceg kandidata delioca
sumaPom(N, S, I):- I=<N//2, N mod I=:=0, I1 is I+1, sumaPom(N, S1, I1), S is S1+I.
```

```
% ako I ne deli, ne dodajemo ga na tekucu sumu vec samo pozivamo za sledeceg
    kandidata delioca
sumaPom(N, S, I):- I=<N//2, N mod I=\=0, I1 is I+1, sumaPom(N, S, I1).

% ako provera prodje, tada se izvrsava predikat sumaDelilaca i proverava da li je
    dobijena suma jednaka broju N, ako provera ne prodje, ispisace se poruka i
    prekinuti program
savrsen(N):- provera(N), sumaDelilaca(N, S), N=:=S.

/*
| ?- savrsen(6)
yes
| ?- savrsen(-6)
Broj nije prirodan
no
*/
```

Rešenje 8.29

```
% mozemo kao u 4. zadatku pisati pomocni predikat ili uslov ispravnosti argumenta
    ubaciti u predikat direktno, za svako N < 1 izbaci3 vraca no i prekida program
    jer ce sve klauze biti netacne

% bazni slucaj kad znamo rezultat
izbaci3(1, 0):- !.

% delimo na slucajeve da li je tekuca cifra manja ili veca/jednaka 3
izbaci3(N, X):- N>1, N mod 10>=3, N1 is N//10, izbaci3(N1, X1), X is X1*10+N mod 10,
    !.
izbaci3(N, X):- N>1, N mod 10<3, N1 is N//10, izbaci3(N1, X).

% ako bi trebalo ispisati poruku da je neispravan ulaz, mozemo dodati klauzu i za N<1
% bez fail, za upit izbaci3(-12,X) daje odgovor yes, a sa fail no
izbaci3(N, _):- N<1, writeln('Neispravan argument N'), fail.
```

Rešenje 8.30

```
% pomocni predikat za proveru pripadnosti elementa listi
sadrzi(X, [X|_]):- !.
sadrzi(X, [G|R]):- G \== X, sadrzi(X, R).

% izbacivanje duplikata iz liste
duplikati([], []).
duplikati([G|R], [G|R1]):- not(sadrzi(G, R)), duplikati(R,R1), !.
duplikati([_|R], R1):- duplikati(R,R1).

% pomocni predikat za spajanje dve liste
spoji([], L, L).
spoji([G|R1], L2, [G|R3]):- spoji(R1, L2, R3).

% unija listi - jedna ideja: spajamo liste pa uklanjamo duplikate
% za vezbu implementirati ovaj predikat tako da se duplikati uklanjaju pri samom
    spajanju listi
unija(L1, L2, L):- spoji(L1, L2, L3), duplikati(L3, L).

% presek listi
presek([], _, []).
presek([G|R1], L2, [G|R3]):- sadrzi(G, L2), presek(R1, L2, R3), !.
presek([_|R1], L2, L):- presek(R1, L2, L).

% razlika listi L1 i L2
razlika([], _, []).
razlika([G1|R1], L2, R3):- sadrzi(G1, L2), razlika(R1, L2, R3), !.
razlika([G|R1], L2, [G|R3]):- razlika(R1, L2, R3).
```

Rešenje 8.31

```
/*
```

```
strukturama oblika k(boja, nacionalnost, jelo, pice, kucniLjubimac) opisujemo date
    cinjenice, a u listi L su kuce poredjane jedna pored druge, tako da po redosledu
    u listi imamo informaciju da li je kuca desno od neke druge kuce i da li su kuce
    jedna pored druge
*/

/*
pomocni predikat koji proverava da li je X clan liste, njemu prosledjujemo strukturu
    k sa poznatim vrednostima iz teksta i opisujemo kakva kuca treba da bude u listi
*/
clan(X, [X|_]).
clan(X, [_|R]):- clan(X,R).

% predikat smesta u listu L kuce koje zadovoljavaju uslove iz teksta, tj. predikat L
    unifikuje sa resenjem zagonetke
% u listu ubacujemo cinjenice koje su vezane za raspored kuca i to samo one koje
    jednoznacno odredjuju poziciju kuce u listi
kuce(L):- L = [ k(_,norvezanin,_,_,_),
    k(plava,_,_,_,_),
    k(_,_,_,mleko,_),
    k(_,_,_,_,_),
    k(_,_,_,_,_) ],
    % dodajemo kakve sve kuce treba da budu u listi
    clan(k(crvena, englez,_,_,_),L),
    clan(k(_, spanac,_,_,pas),L),
    clan(k(zelena,_,_,kafa,_),L),
    clan(k(_, ukrajinac,_,caj,_),L),
    % kada informacija daje relaciju za neke dve kuce iz liste koristimo predikate
    desno i pored
    desno(k(zelena,_,_,kafa,_),k(bela,_,_,_,_),L),
    clan(k(_,_,spagete,_,puz),L),
    clan(k(zuta,_,pica,_,_),L),
    pored(k(_,_,piletina,_,_),k(_,_,_,_,lisica),L),
    pored(k(_,_,pica,_,_),k(_,_,_,_,konj),L),
    clan(k(_,_,brokoli,narandza,_),L),
    clan(k(_,japanac,susi,_,_),L),
    % medju datim informacijama se ne pominje zebra niti voda, ali posto je krajnje
    pitanje vezano za ove pojmove, moramo dodati da takvi clanovi treba da postoje u
    resenju zagonetke
    clan(k(_,_,_,_,zebra),L),
    clan(k(_,_,_,voda,_),L).

% proverava da li su kuce X i Y jedna pored druge u listi L
pored(X,Y,[X,Y|_]).
pored(X,Y,[Y,X|_]).
pored(X,Y,[_|R]):- pored(X,Y,R).

% proverava da li je kuca X desno od kuce Y u listi L
desno(X,Y,[Y,X|_]).
desno(X,Y,[_|R]):- desno(X,Y,R).

% predikat zagonetka daje odgovor na pitanje cija je zebra, a ko pije vodu, tako sto
    prvo trazi resenje zagonetke pa iz njega izdvaja samo potrebne clanove
zagonetka(X,Y):- kuce(L), clan(k(_,X,_,_,zebra),L), clan(k(_,Y,_,voda,_),L) .

/*
resenje:

| ?- kuce(L)
L = [k(zuta,norvezanin,pica,voda,lisica),k(plava,ukrajinac,piletina,caj,konj),k(
    crvena,englez,spagete,mleko,puz),k(bela,spanac,brokoli,narandza,pas),k(zelena,
    japanac,susi,kafa,zebra)]

odgovor na pitanje:

| ?- zagonetka(X,Y)
X = japanac
Y = norvezanin

*/
```

Rešenje 8.32

```
/*
resenje je lista L sa 5 struktura oblika: d(ime, prezime, godine)
*/

% predikat kojim se proverava da li je X clan liste
clan(X, [X|_]).
clan(X, [_|R]):- clan(X,R).
% predikat smesta u listu L decu koja zadovoljavaju uslove iz teksta
% redosled dece u resenju nije bitan
% direktno u listi mozemo naznaciti ili broj godina, ili data imena ili data
    prezimena, ovde su izabrane godine, a ostale informacije dodate preko predikata
    clan
deca(L):- L = [d(_,_,2), d(_,_,3), d(_,_,4), d(_,_,5), d(_,_,6)],
    clan(d(lazar,jankovic,_),L),
    clan(d(kata,_,G1),L),
    clan(d(_,ivanovic,G2),L),
    clan(d(nevenka,_,G3),L),
    clan(d(_,filipovic,G4),L),
    clan(d(marko,_,G5),L),
    clan(d(ognjen,_,G6),L),
    clan(d(_,hadzic,G7),L),
    clan(d(_,grbovic,_),L),
    % medju informacijama imamo odnos izmedju broja godina odredjene dece, ali ne
    znamo tacno koliko godina imaju ta deca, zbog toga koristimo promenljive kojima
    opisujemo date relacije, obratiti paznju da je potrebno koristiti aritmeticko
    poredjenje =:=, a ne poredjenje na identicnost (==), jer ne zelimo da se porede
    termovi oblika Gi i Gj+N vec njihove brojevne vrednosti prilikom izgradjivanja
    resenja
    G1=:=G2+1, G2=:=G3+1, G4=:=G5+3, G6=:=2*G7.

/*
resenje:

| ?- deca(L)
L = [d(marko,hadzic,2),d(nevenka,grbovic,3),d(ognjen,ivanovic,4),d(kata,filipovic,5),
    d(lazar,jankovic,6)] ?yes


*/
```

9

Programiranje ograničenja - Prolog

Potrebno je imati instaliran B-Prolog na računaru.

Literatura:

(a) http://www.picat-lang.org/bprolog/

(b) http://www.picat-lang.org/bprolog/download/manual.pdf

9.1 Programiranje ograničenja

9.1.1 Uvodni primeri

Zadatak 9.1 Osnovni pojmovi i postavka problema.

```
/*

Programiranje ogranicenja nad konacnim domenom:
1) generisanje promenljivih i njihovih domena
2) generisanje ogranicenja nad promenljivima
3) instanciranje promenljivih ili obelezavanje

Definisanje domena (D) promenljivih:
-- X in D ili X :: D -> promenljiva X uzima samo vrednosti iz konacnog domena D
-- Vars in D ili Vars :: D -> sve promenjive iz liste Vars uzimaju samo vrednosti iz
    konacnog domena D
Domen se definise kao interval u obliku Pocetak..Korak..Kraj (Korak je opcion i
    ukoliko se ne navede, podrazumeva se da je Korak = 1)
Primeri za domen:
1..10  -> 1,2,3,4,5,6,7,8,9,10
1..2..10 -> 1,3,5,7,9

Osnovni predikati za ogranicenja

1) opsta:
-- alldifferent(Vars) ili alldistinct(Vars) - sve vrednosti razlicite u listi termova
     Vars
-- atmost(N,L,V) - najvise N elemenata iz skupa L jednako sa V (N mora biti Integer,
    V term, a L lista termova)
-- atleast(N,L,V) - najmanje N elemenata iz skupa L jednako sa V (N mora biti Integer
    , V term, a L lista termova)
-- excatly(N,L,V) - tacno N elemenata iz skupa L jednako sa V (N mora biti Integer, V
     term, a L lista termova)

2) aritmeticka:
-- E1 R E2 gde su E1 i E2 aritmeticki izrazi, a R iz skupa {#=, #\=, #>=, #>, #=<,
    #<}
-- min(L) - minimalni element iz liste termova L
-- max(L) - maksimalni element iz liste termova L
```

```
-- max(E1, E2)/min(E1, E2) -- manji/veci od izraza E1 i E2
-- sum(L) - suma elemenata liste termova L

Instanciranje i prikazivanje promenljivih: labeling(Vars).
Funkcija labeling se moze pozvati i sa razlicitim opcijama u obliku labeling(Options,
     Vars)
gde je Options lista opcija. Ukoliko se pozove sa labeling(Vars), podrazumevano je
    Options = []. Neke od opcija:
-- minimize(E) - trazi instance za Vars pri kojima je vrednost celobrojnog izraza E
    minimalna
-- maximize(E) - trazi instance za Vars pri kojima je vrednost celobrojnog izraza E
    maksimalna

*/

/*
Primer: X pripada skupu {1,2,3}, Y skupu {2,4,6,8,10}, Z skupu {5,6,7,8} i vazi Z>=Y
*/

primer(Vars) :- Vars = [X, Y, Z], % generisanje promenljivih
    X :: 1..3,    % definisanje domena
    Y :: 2..2..10,
    Z :: 5..8,
    Z #>= Y,    % ogranicenje
    labeling(Vars).   % instanciranje

/*
| ?- primer(Vars).
Vars = [1,2,5] ?;
Vars = [1,2,6] ?;
Vars = [1,2,7] ?;
Vars = [1,2,8] ?;
Vars = [1,4,5] ?;
Vars = [1,4,6] ?;
Vars = [1,4,7] ?;
Vars = [1,4,8] ?;
Vars = [1,6,6] ?;
Vars = [1,6,7] ?;
Vars = [1,6,8] ?;
Vars = [1,8,8] ?;
...
*/

/*
Primer: ispisati sve brojeve od 1..100 koji su puni kvadrati
 */

puni(Vars) :- Vars = [X],
           Vars :: 1..100,
           Y*Y #= X ,
           labeling(Vars).

/*
| ?- puni(Vars)
Vars = [1] ?;
Vars = [4] ?;
Vars = [9] ?;
Vars = [16] ?;
Vars = [25] ?;
Vars = [36] ?;
Vars = [49] ?;
Vars = [64] ?;
Vars = [81] ?;
Vars = [100] ?;
no

*/
```

9.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 9.2 Napisati program koji pronalazi petocifren broj `ABCDE` za koji je izraz `A+2*B-`

`3*C+4*D-5*E` minimalan i `A`, `B`, `C`, `D` i `E` su različite cifre.

[Rešenje 9.2]

Zadatak 9.3 Dati su novčići od 1, 2, 5, 10, 20 dinara. Napisati program koji pronalazi sve moguće kombinacije tako da zbir svih novčića bude 50 i da se svaki novčič pojavljuje bar jednom u kombinaciji.

[Rešenje 9.3]

Zadatak 9.4 Napisati program koji ređa brojeve u magičan kvadrat. Magičan kvadrat je kvadrat dimenzija 3x3 takav da je suma svih brojeva u svakom redu, svakoj koloni i svakoj dijagonali jednak 15 i svi brojevi različiti. Na primer:

```
4 9 2
3 5 7
8 1 6
```

[Rešenje 9.4]

Zadatak 9.5 Napisati program koji pronalazi sve vrednosti promenljivih X, Y, Z za koje važi da je $X >= Z$ i $X * 2 + Y * X + Z <= 34$ pri čemu promenljive pripadaju narednim domenima $X \in \{1, 2, ..., 90\}$, $Y \in \{2, 4, 6, ...60\}$ i $Z \in \{1, 10, 20, ..., 100\}$

[Rešenje 9.5]

Zadatak 9.6 Napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
 TWO
+TWO
---------
FOUR
```

[Rešenje 9.6]

Zadatak 9.7 Napisati program koji pronalazi sve vrednosti promenljivih X, Y, Z i W za koje važi da je $X >= 2 * W$, $3 + Y <= Z$ i $X - 11 * W + Y + 11 * Z <= 100$ pri čemu promenljive pripadaju narednim domenima $X \in \{1, 2, ..., 10\}$, $Y \in \{1, 3, 5, ...51\}$, $Z \in \{10, 20, 30, ..., 100\}$ i $W \in \{1, 8, 15, 22, ..., 1000\}$.

[Rešenje 9.7]

Zadatak 9.8 Napisati program koji raspoređuje brojeve 1-9 u dve linije koje se seku u jednom broju. Svaka linija sadrži 5 brojeva takvih da je njihova suma u obe linije 25 i brojevi su u rastućem redosledu.

```
1   3
 2 4
  5
 6 8
7   9
```

[Rešenje 9.8]

Zadatak 9.9 Pekara *Kiflica* proizvodi hleb i kifle. Za mešenje hleba potrebno je 10 minuta, dok je za kiflu potrebno 12 minuta. Vreme potrebno za pečenje ćemo zanemariti. Testo za hleb sadrži 300g brašna, a testo za kiflu sadrži 120g brašna. Zarada koja se ostvari prilikom prodaje jednog hleba je 7 dinara, a prilikom prodaje jedne kifle je 9 dinara. Ukoliko pekara ima 20 radnih sati za mešenje peciva i 20kg brašna, koliko komada hleba i kifli treba da se umesi kako bi se ostvarila maksimalna zarada (pod pretpostavkom da će pekara sve prodati)?

[Rešenje 9.9]

Zadatak 9.10 Napisati program pronalazi vrednosti A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S (svako slovo predstavlja različit broj) koje su poređane u heksagon na sledeći način:

```
  A,B,C
 D,E,F,G
H,I,J,K,L
 M,N,O,P
  Q,R,S
```

tako da zbir vrednosti duž svake horizontalne i dijagonalne linije bude 38 (A+B+C = D+E+F+G = ... = Q+R+S = 38, A+D+H = B+E+I+M = ... = L+P+S = 38, C+G+L = B+F+K+P = ... = H+M+Q = 38).

[Rešenje 9.10]

Zadatak 9.11 Kompanija Start ima 250 zaposlenih radnika. Rukovodstvo kompanije je odlučilo da svojim radnicima obezbedi dodatnu edukaciju. Da bi se radnik obučio programskom jeziku Elixir potrebno je platiti 100 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 150 projekat/sati mesečno, što bi za kompaniju značilo dobit od 5 evra po projekat/satu. Da bi se radnik obučio programskom jeziku Dart potrebno je platiti 105 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 170 projekat/sati mesečno, koji bi za kompaniju značili dobit od 6 evra po satu. Ukoliko Start ima na raspolaganju 26000 evra za obuku i maksimalan broj 51200 mogućih projekat/sati mesečno, odrediti na koji nacin kompanija treba da obuči svoje zaposlene kako bi ostvarila maksimalnu dobit.

[Rešenje 9.11]

Zadatak 9.12 Napisati program koji raspoređuje n dama na šahovsku tablu dimenzije nxn tako da se nikoje dve dame ne napadaju.

[Rešenje 9.12]

Zadatak 9.13 Napisati program koji za ceo broj n ispisuje magičnu sekvencu S brojeva od 0 do $n-1$. $S = (x_0, x_1, ..., x_{n-1})$ je magična sekvenca ukoliko postoji x_i pojavljivanja broja i za $i = 0, 1, ..., n-1$.

[Rešenje 9.13]

9.1.3 Zadaci za vežbu

Zadatak 9.14 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
GREEN + ORANGE = COLORS
MANET + MATISSE + MIRO + MONET + RENOIR = ARTISTS
COMPLEX + LAPLACE = CALCULUS
THIS + IS + VERY = EASY
CROSS + ROADS = DANGER
FATHER + MOTHER = PARENT
WE + WANT + NO + NEW + ATOMIC = WEAPON
EARTH + AIR + FIRE + WATER = NATURE
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS
SEE + YOU = SOON
NO + GUN + NO = HUNT
WHEN + IN + ROME + BE + A = ROMAN
DONT + STOP + THE = DANCE
HERE + THEY + GO = AGAIN
OSAKA + HAIKU + SUSHI = JAPAN
MACHU + PICCHU = INDIAN
SHE + KNOWS + HOW + IT = WORKS
COPY + PASTE + SAVE = TOOLS
```

Zadatak 9.15 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da suma bude zadovoljena:

```
THREE + THREE + ONE = SEVEN
NINE + LESS + TWO = SEVEN
ONE + THREE + FOUR = EIGHT
THREE + THREE + TWO + TWO + ONE = ELEVEN
SIX + SIX + SIX = NINE + NINE
SEVEN + SEVEN + SIX = TWENTY
ONE + ONE + ONE + THREE + THREE + ELEVEN = TWENTY
EIGHT + EIGHT + TWO + ONE + ONE = TWENTY
ELEVEN + NINE + FIVE + FIVE = THIRTY
NINE + SEVEN + SEVEN + SEVEN = THIRTY
TEN + SEVEN + SEVEN + SEVEN + FOUR + FOUR + ONE = FORTY
TEN + TEN + NINE + EIGHT + THREE = FORTY
FOURTEEN + TEN + TEN + SEVEN = FORTYONE
NINETEEN + THIRTEEN + THREE + TWO + TWO + ONE + ONE + ONE = FORTYTWO
FORTY + TEN + TEN = SIXTY
SIXTEEN + TWENTY + TWENTY + TEN + TWO + TWO = SEVENTY
SIXTEEN + TWELVE + TWELVE + TWELVE + NINE + NINE = SEVENTY
TWENTY + TWENTY + THIRTY = SEVENTY
FIFTY + EIGHT + EIGHT + TEN + TWO + TWO = EIGHTY
FIVE + FIVE + TEN + TEN + TEN + TEN + THIRTY = EIGHTY
SIXTY + EIGHT + THREE + NINE + TEN = NINETY
ONE + NINE + TWENTY + THIRTY + THIRTY = NINETY
```

Zadatak 9.16 Za svaku narednu zagonetku, napisati program koji dodeljuje različite vrednosti različitim karakterima tako da jednakost bude zadovoljena:

```
MEN * AND = WOMEN
COGITO = ERGO * SUM
((JE + PENSE) - DONC) + JE = SUIS
FERMAT * S = LAST + THEOREM.
WINNIE / THE = POOH
TWO * TWO + EIGHT = TWELVE
```

Zadatak 9.17 Uraditi sve zadatke koji su pobrojani ovde:
http://www.primepuzzle.com/leeslatest/alphameticpuzzles.html

Zadatak 9.18 Čistačica Mica sređuje i čisti kuće i stanove. Da bi sredila i počistila jedan stan potrebno joj je 1 sat, dok joj je za kuću potrebno 1.5 sati. Prilikom čišćenja, Mica potroši neku količinu deterdženta, 120ml po stanu, odnosno 100ml po kući. Mica zaradi 1000 dinara po svakom stanu, odnosno 1500 dinara po kući. Ukoliko Mica radi 40 sati nedeljno i ima 5l deterdženta na raspolaganju, koliko stanova i kuća je potrebno da očisti kako bi imala najveću zaradu?

Zadatak 9.19 Marija se bavi grnčarstvom i pravi šolje i tanjire. Da bi se napravila šolja, potrebno je 6 minuta, dok je za tanjir potrebno 3 minuta. Pri pravljenju šolje potroši se 75 gr, dok se za tanjir potroši 100 gr gline. Ukoliko ima 20 sati na raspolaganju za izradu svih proizvoda i 250 kg gline, a zarada koju ostvari iznosi 2 evra po svakoj šolji i 1.5 evra po tanjiru, koliko šolja i tanjira treba da napravi kako bi ostvarila maksimalnu zaradu?

Zadatak 9.20 Jovanin komšija preprodaje računare i računarsku opremu. Očekuje isporuku računara i štampača. Pri tom, računari su spakovani tako da njihova kutija zauzima 360 kubnih decimetara prostora, dok se štampači pakuju u kutijama koje zauzimaju 240 kubnih decimetara prostora. Komšija se trudi da mesečno proda najmanje 30 računara i da taj broj bude bar za 50% veći od broja prodatih štampača. Računari koštaju 200 evra po nabavnoj ceni, a prodaju se po ceni od 400 evra, dok štampači koštaju u nabavci 60 evra i prodaju se za 140 evra. Magacin kojim komšija raspolaže ima svega 30000 kubnih decimetara prostora i mesečno može da nabavi robu u iznosu od najviše 14000 evra. Koliko računara, a koliko štampača komšija treba da proda kako bi se maksimalno obogatio?

9.2 Rešenja

Rešenje 9.2

```
% funkciji labeling prosledjujemo zahtev za minimizaciju trazenog izraza u listi
petocifren :-   Vars = [A,B,C,D,E],
    % definisemo domen
    A :: 1..9,
    B :: 0..9,
    C :: 0..9,
    D :: 0..9,
    E :: 0..9,
    % dodajemo uslov
    alldifferent(Vars),
    % prilikom obelezavanja prosledjujemo i uslov minimizacije
    labeling([minimize(A+2*B-3*C+4*D-5*E)],Vars),
    % prevodimo dobijene vednosti u broj
    Broj is 10000*A+1000*B+100*C+10*D+E,
    % ispisujemo resenje
    write(Broj), nl.


```

Rešenje 9.3

```
% promenljive oznacavaju broj novcica u kombinaciji redom za A - 1 din, B - 2 din, C
    - 5 din, D - 10 din, E - 20 din.
kombinacije(Vars) :- Vars = [A,B,C,D,E],
    % definisemo domen
    A :: 1..50,
    B :: 1..25,
    C :: 1..10,
    D :: 1..5,
    E :: 1..2,
    % dodajemo uslov
    A+2*B+5*C+10*D+20*E #= 50,
    labeling(Vars),
    % ispisujemo resenje
    write(A+2*B+5*C+10*D+20*E = 50), nl.
```

Rešenje 9.4

```
magicni(Vars):- Vars=[X1,X2,X3,X4,X5,X6,X7,X8,X9],
    % domen za sve je isti
    Vars :: 1..9,
    % uslov razlicitosti
    alldifferent(Vars),
    % uslovi za zbirove
    X1+X2+X3#=15,
    X4+X5+X6#=15,
    X7+X8+X9#=15,
        X1+X4+X7#=15,
        X2+X5+X8#=15,
        X3+X6+X9#=15,
        X1+X5+X9#=15,
        X3+X5+X7#=15,
        labeling(Vars),
    write(X1), write(' '), write(X2), write(' '), write(X3), nl,
    write(X4), write(' '), write(X5), write(' '), write(X6), nl,
    write(X7), write(' '), write(X8), write(' '), write(X9), nl.
/*
| ?- magicni(Vars).
2 7 6
9 5 1
```

```
4 3 8
Vars = [2,7,6,9,5,1,4,3,8] ?;
2 9 4
7 5 3
6 1 8
Vars = [2,9,4,7,5,3,6,1,8] ?;
4 3 8
9 5 1
2 7 6
...
*/
```

Rešenje 9.5

```
pronadji(Vars):- Vars=[X,Y,Z],
    % definisemo domen
    X :: 1..90,
    Y :: 2..2..60,
    Z :: 1..10..100,
    % definisemo ogranicenja
    Z #=< X,
    2*X+Y*X+Z #=< 34,
    % instanciramo promenljive
    labeling(Vars).

/*
| ?- pronadji(Vars)
Vars = [1,2,1] ?;
Vars = [1,4,1] ?;
Vars = [1,6,1] ?;
Vars = [1,8,1] ?;
Vars = [1,10,1] ?;
...

*/
```

Rešenje 9.6

```
jednakost(Vars):- Vars=[T,W,O,F,U,R],
    Vars :: 0..9,
    % iskljucujemo za T i F nulu iz domena
    T#\=0,
    F#\=0,
    alldifferent(Vars),
    2*(T*100+W*10+O) #= F*1000+O*100+U*10+R,
    labeling(Vars),
    write(' '), write(T), write(W), write(O), nl,
    write('+'), write(T), write(W), write(O), nl,
    write('----------'), nl,
    write(F), write(O), write(U), write(R), nl.
/*
| ?- jednakost(Vars)
 734
+734
----------
1468
Vars = [7,3,4,1,6,8] ?;
 765
+765
----------
1530
...

*/

/*
BITNO: Ukoliko se na kraju predikata doda fail, ispisace se sva resenja pri pozivu,
    inace staje cim pronadje jedno resenje, pa sledece dobijamo kucanjem ; za
    nastavljanje pretrage
kao u prethodnom primeru
```

```
*/

jednakostSvi(Vars):- Vars=[T,W,O,F,U,R],
    Vars :: 0..9,
    % iskljucujemo za T i F nulu iz domena
    T#\=0,
    F#\=0,
    alldifferent(Vars),
    2*(T*100+W*10+O) #= F*1000+O*100+U*10+R,
    labeling(Vars),
    write(' '), write(T), write(W), write(O), nl,
    write('+'), write(T), write(W), write(O), nl,
    write('------'), nl,
    write(F), write(O), write(U), write(R), nl,
    % odvajamo resenja jer zbog fail ne staje kod prvog
    % resenja vec ce nastaviti pretragu
    write('----------'), nl, fail.
```

Rešenje 9.7

```
pronadji(Vars):- Vars=[X,Y,Z,W],
    % definisemo domen
    X :: 1..10,
    Y :: 1..2..51,
    Z :: 10..10..100,
    W :: 1..7..1002,
    % definisemo ogranicenja
    2*W #=< X,
    3+Y #=< Z,
    X-11*W+Y+11*Z #=< 100,
    % instanciramo promenljive
    labeling(Vars).
/*
Ovo je primer sistema nejednacina bez resenja:

| ?- pronadji(Vars)
no

*/
```

Rešenje 9.8

```
% A1, B1, C1, D1, E1 predstavljaju brojeve na glavnoj dijagonali
% A2, B2, C1, D2, E2 predstavljaju brojeve na glavnoj dijagonali
dijagonale(Vars):- Vars=[A1,B1,C1,D1,E1,A2,B2,D2,E2],
    % domen za sve je isti
    Vars :: 1..9,
    % uslov razlicitosti
    alldifferent(Vars),
    % uslovi za zbirove
    A1+B1+C1+D1+E1#=25,
    A2+B2+C1+D2+E2#=25,
    % uslovi za poredak
    A1#<B1, B1#<C1, C1#<D1, D1#<E1,
    A2#<B2, B2#<C1, C1#<D2, D2#<E2,
        labeling(Vars),
    write(A1), write('     '), write(A2), nl,
    write(' '), write(B1), write('   '), write(B2), nl,
    write('   '), write(C1), nl,
    write(' '), write(D2), write('   '), write(D1), nl,
    write(E2), write('     '), write(E1), nl.
```

Rešenje 9.9

```
/*
 H - broj komada hleba, K - broj komada kifli

 H>=0
 K>=0
```

```
 S obzirom na to da imamo 20kg brasna na raspolaganju, mozemo napraviti:
 - najvise 20000/120 kifli
 - najvise 20000/300 hleba

 K <= 20000/120 ~ 166
 H <= 20000/300 ~ 66

 S obzirom na to da imamo 20h na raspolaganju, mozemo napraviti:
 - najvise 1200/12 kifli
 - najvise 1200/10 hleba

 H <= 1200/10 = 120
 K <= 1200/12 = 100

 najoptimalnije je za gornju granicu domena postaviti minimum od dobijenih vrednosti,
     tj. sve ukupno H <= 66, K <= 100

*/

pekara(Vars) :- Vars = [H, K],
    H :: 0..66,
    K :: 0..100,

/*
 Ogranicenja vremena:
 - vreme potrebno za mesenje jednog hleba je 10 min,
   tj. za mesenje H komada hleba potrebno je 10*H minuta
 - vreme potrebno za mesenje jedne kifle je 12 min,
   tj. za mesenje K komada kifli potrebno je 12*K minuta

 Ukupno vreme koje je na raspolaganju iznosi 20h, tako da je:
 10*H + 12*K <= 1200
*/

    10*H + 12*K #=< 1200,

/*
 Ogranicenje materijala:
 - za jedan hleb potrebno je 300g brasna, a za H komada hleba potrebno je H*300 grama
 - za jednu kifli potrebno je 120g brasna, a za K komada kifli potrebno je K*120
    grama

 Ukupno, na raspolaganju je 20kg brasna, tako da je:
 300*H + 120*K <= 20000
*/
    300*H + 120*K #=< 20000,


/*
 Zarada iznosi:
 - 7din/hleb, tj. zarada za H komada hleba bice 7*H
 - 9din/kifla tj. zarada za K komada kifli bice 9*K

 Ukupna zarada iznosi:
 7*H + 9*K - funkcija koju treba maksimizovati - ovo dodajemo prilikom obelezavanja
*/
    labeling([maximize(7*H+9*K)], Vars),
    Zarada is 7*H+9*K,
    write('Maksimalna zarada od '), write(Zarada), write(' dinara se ostvaruje za '),
    write(H), write(' komada hleba i '), write(K), write(' komada kifli.'), nl.
```

Rešenje 9.10

```
heksagon(Vars):- Vars=[A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S],
    % domen za sve je isti
    Vars :: 1..38,
    % uslov razlicitosti
    alldifferent(Vars),
/*
 Dodajemo ogranicenja za svaku horizontalnu liniju
  A,B,C
```

```
 D,E,F,G
H,I,J,K,L
 M,N,O,P
  Q,R,S
*/
    A+B+C#=38,
    D+E+F+G#=38,
    H+I+J+K+L#=38,
        M+N+O+P#=38,
        Q+R+S#=38,
% Dodajemo ogranicenja za svaku od glavnih dijagonala
    H+M+Q#=38,
    D+I+N+R#=38,
    A+E+J+O+S#=38,
        B+F+K+P#=38,
        C+G+L#=38,
% Dodajemo ogranicenja za svaku od sporednih dijagonala
    A+D+H#=38,
    B+E+I+M#=38,
    C+F+J+N+Q#=38,
        G+K+O+R#=38,
        L+P+S#=38,

        labeling(Vars),
    write('  '), write(A), write(' '), write(B), write(' '), write(C), nl,
    write(' '), write(D),write(' '), write(E), write(' '),write(F),write(' '), write(
    G), nl,
    write(H), write(' '),write(I),write(' '), write(J),write(' '), write(K),write(' '
    ), write(L), nl,
    write(' '), write(M),write(' '), write(N), write(' '),write(O),write(' '), write(
    P), nl,
    write('  '), write(Q),write(' '), write(R), write(' '),write(S), nl.
```

Rešenje 9.11

```
/*
kompanija ima 250 zaposlenih radnika
za sve njih organizuje dodatnu obuku
ako je E promenjiva za Elixir, a D za Dart
mora da vazi (gruba procena) E<=250, D<=250 i E + D = 250
*/

kompanija(Vars) :- Vars = [E, D],
    Vars :: 0..250,

    % dodajemo uslov za ukupan broj radnika
    E+D#=250,

    % dodajemo ogranicenje za broj projekat sati
    150*E+170*D#=<51200,

    % dodajemo ogranicenje za raspoloziva sredstva
    100*E+105*D#=<26000,

/*
ukupna zarada se dobija kada od ostvarene dobiti preko broja projekat/sati oduzmemo
    gubitak za placanje kurseva radnicima, tj. funkcija koju treba maksimizovati je:

150*5*E + 170*6*D - (100*E + 105*D) --> ovo dodajemo kod obelezavanja
*/
    labeling([maximize(150*5*E + 170*6*D - (100*E + 105*D))],Vars),
    Zarada is (150*5*E + 170*6*D - (100*E + 105*D)),
    write('Maksimalna zarada je '), write(Zarada),
    write(', broj radnika koje treba poslati na kurs Elixir je '), write(E),
    write(', a broj radnika koje treba poslati na kurs Dart je '), write(D),nl.
```

Rešenje 9.12

```
/*
```

```
Kraljica se moze pomerati u pravoj liniji vertikalno, horizontalno ili dijagonalno,
    za bilo koliko slobodnih polja. Resenje zahteva da nikoje dve dame ne dele istu
    vrstu, kolonu ni dijagonalu. Problem je opstiji od poznatog problema osam
    sahovskih dama jer u ovom problemu treba N dama postaviti na sahovsku tablu
    dimenzije NxN koje se zadaje kao ulazni podatak.

*** U resenju se koristi petlja FOREACH. Opsti oblik:

  foreach(E1 in D1, . . ., En in Dn, LocalVars, Goal)

- Ei - moze biti bilo koji term, najcesce promenljiva
- Di - lista ili opseg celih brojeva oblika L..R (L i R su ukljuceni)
- LocalVars (opciono) - lista lokalnih promenljivih
- Goal - predikat koji se poziva

Primeri:

| ?- foreach(I in [1,2,3], write(I))
123
yes

Domen za X i Y u ovom slucaju mora biti iste kardinalnosti
| ?- foreach((X,Y) in ([a,b], 1..2), writeln(X==Y))
a==1
b==2
yes

| ?- foreach(X in [a,b], Y in [1,2], writeln(X==Y))
a==1
a==2
b==1
b==2
yes
*/

% Indeksi niza Qs I i J predstavljaju kolone u kojima su kraljice, a elementi niza Qs
    [I] i Qs[J] predstavljaju vrste u kojima se nalaze kraljice
kraljice(N):- length(Qs,N),   % Qs je niz, tj. lista od n promenljivih
    Qs :: 1..N,
% I je implicitno razlicito od J -> razlicite kolone
% Qs[I] treba da bude razlicito od Qs[J] -> razlicite vrste
% apsolutna vrednost razlike vrsta treba da bude razlicita od apsolutne vrednosti
    razlike kolona -> razlicite dijagonale
    foreach(I in 1..N-1, J in I+1..N,
    (Qs[I] #\= Qs[J], abs(Qs[I]-Qs[J]) #\= J-I)),
    labeling(Qs),
    writeln(Qs), fail.
% stavljanjem predikata fail na kraj, dobicemo sve moguce kombinacije za kraljice
% bez fail, program staje posle prvog pronadjenog rezultata
```

Rešenje 9.13

```
/*
Primer magicne sekvence za N=5:
[2,1,2,0,0]

Izvrsavanje programa za gornji primer (1 sa desne strane kad je uslov S[J]#=I
    ispunjen, 0 kad nije):
sum([(S[J]#=I) : J in 1..N])#=S[I+1])
I=0
   S[1]==0 0
   S[2]==0 0
   S[3]==0 0
   S[4]==0 1
   S[5]==0 1
   ------
   ukupno 2
   S[1]==2 tacno! (imamo dve nule)
I=1
   S[1]==1 0
   S[2]==1 1
   S[3]==1 0
```

```
   S[4]==1 0
   S[5]==1 0
   ------
   ukupno 1
   S[2]==1 tacno! (imamo jednu jedinicu)
I=2
   S[1]==2 1
   S[2]==2 0
   S[3]==2 1
   S[4]==2 0
   S[5]==2 0
   ------
   ukupno 1
   S[3]==2 tacno! (imamo dve dvojke)
I=3
   S[1]==3 0
   S[2]==3 0
   S[3]==3 0
   S[4]==3 0
   S[5]==3 0
   ------
   ukupno 0
   S[4]==0 tacno! (imamo 0 trojki)

I=4
   S[1]==4 0
   S[2]==4 0
   S[3]==4 0
   S[4]==4 0
   S[5]==4 0
   ------
   ukupno 0
   S[5]==0 tacno! (imamo 0 cetvorki)

*/

/*

Zadavanje listi u obliku: [T : E1 in D1, . . ., En in Dn, LocalVars, Goal]
- za svaku kombinacija vrednosti E1,...,En, ako predikat Goal uspe, T se
dodaje u listu
- LocalVars i Goal su opcioni argumenti

*/
magicna(N):- length(S,N),
    S :: 0..N-1,
/* dodajemo ogranicenja:
element na poziciji S[I+1] treba da bude jednak broju elemenata liste koji su jednaki
     sa I, taj broj dobijamo sumiranjem liste ciji su elementi poredjenja na
    jednakost elementa liste sa trazenim brojem I
*/
    foreach(I in 0..N-1,
    sum([(S[J] #= I) : J in 1..N]) #= S[I+1]),
    labeling(S),
    writeln(S), fail. % da bismo dobili sve, a ne samo jednu magicnu sekvencu
    dodajemo fail predikat na kraj
```