

Programski jezici

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

**Univerzitet u Beogradu
Matematički fakultet**

Programske paradigme

Materijali za vežbe

**Nastavnik: Milena Vujošević Janičić
Asistent: Branislava Živković**

**Beograd
2016.**

Priprema materijala:

dr Milena Vujošević Jančić, docent na Matematičkom fakultetu u Beogradu

Marjana Šolajić, asistent na Matematičkom fakultetu u Beogradu

Branislava Živković, asistent na Matematičkom fakultetu u Beogradu

Sadržaj

1 Skript programiranje	3
1.1 Uvod, kolekcije, matematičke funkcije	3
1.1.1 Uvodni primeri	3
1.1.2 Zadaci za samostalni rad sa rešenjima	7
1.1.3 Zadaci za vežbu	8
1.2 Datoteke, niske, JSON format, datum	8
1.2.1 Uvodni primeri	8
1.2.2 Zadaci za samostalni rad sa rešenjima	10
1.2.3 Zadaci za vežbu	11
1.3 Argumenti komandne linije, sortiranje, obilazak direktorijuma	12
1.3.1 Uvodni primeri	12
1.3.2 Zadaci za samostalni rad sa rešenjima	14
1.3.3 Zadaci za vežbu	15
1.4 Rešenja	15
2 Programiranje ograničenja - Python	21
2.1 Programiranje ograničenja	21
2.1.1 Uvodni primeri	21
2.1.2 Zadaci za samostalni rad sa rešenjima	22
2.1.3 Zadaci za vežbu	24
2.2 Rešenja	25
3 Funkcionalno programiranje	35
3.1 Uvod	35
3.1.1 Uvodni primeri	35
3.1.2 Zadaci za samostalni rad sa rešenjima	37
3.1.3 Zadaci za vežbu	38
3.2 Liste	38
3.2.1 Uvodni primeri	38
3.2.2 Zadaci za samostalni rad sa rešenjima	39
3.2.3 Zadaci za vežbu	40
3.3 Rešenja	40

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/>

1.1 Uvod, kolekcije, matematičke funkcije

1.1.1 Uvodni primeri

Zadatak 1.1 Ispisivanje pozdravne poruke, komentari.

```
1 # Ovako se pisu komentari
#
3 # Pokretanje programa iz terminala:
# $python hello.py
5 #
print "Hello world! :)"
```

Zadatak 1.2 Promenljive, niske, formatiran ispis, učitavanje sa standardnog ulaza, aritmetičke i logičke operacije, naredbe grananja.

```
2 # Promenljive se dinamički tipiziraju
a = 45
4 b = 67.45
istina = True
6 # Niske su konstantne tj. nisu promenljive.
# To znači da se menjanjem nekog karaktera u niski
8 # pravi nova niska u memoriji.
niska = "I believe i can fly!"
10
# Ispis na standardni izlaz
12 print a
print b
14 print a, b, istina

16 # Formatiran ispis
print "\n-----Formatiran ispis-----\n"
18 print "Ceo broj: {0:d} \nBroj u pokretnom zarezu {1:f}\nBulovska vrednost: {2:b}\nNiska: {3:s}\n".format(a,b,istina,niska)

20
# Učitavanje niske sa standardnog ulaza
22 print "\n-----Učitavanje sa standardnog ulaza-----\n"
string_broj = raw_input("Unesite ceo broj: ")
24 broj = int(string_broj) # vrsi se konverzija stringa u ceo broj, slicno: float, str
```

```
26 # Osnovne aritmetičke operacije:
# +, -, *, /, %, ** (stepenovanje)
28 print "\n-----Osnovne aritmetičke operacije-----\n"
print broj+4
30
# Osnovne logičke operacije:
32 # not, and, or
print "\n-----Osnovne logičke operacije-----\n"
34 print istina or False
36
# Blokovi se ne ograničavaju viticastim zagradama kao što je u C-u
# već moraju biti uvučeni tabulatorom.
38
# Naredba grananja
40 print "\n-----Naredba grananja-----\n"
if broj%2 == 0:
42     print "Unet je paran broj \n"
elif broj%3 == 0:
44     print "Unet je broj deljiv sa 3\n"
# Naredbi <<elif>> može biti više
46 else:
    print "Unet je broj koji nije ni paran ni deljiv sa 3\n"
48
# Naredba <<switch>> ne postoji
50
# Petlja
52 print "\n-----Petlja <<while>>-----\n"
i=1
54 while i<=10:
    print i
56     i=i+1 # i++ ne postoji, može ili ovako ili i+=1
58
# Naredba <<break>> iskace iz bloka, isto kao i u C-u
# Naredba <<pass>> je ista kao naredba <<continue>> u C-u
60
# Funkcije
62 #
# def ime_funkcije(argumenti):
64 #     telo funkcije
#
66
def f1(x,y):
68     return x+y
70
print f1(2,2)
72
def f2(a):
    return [a,2*a,3*a]
74
print f2(11.1)
```

Zadatak 1.3 Moduli math i random.

```
1 # Matematičke funkcije
3 # Uključujemo modul <<math>>
import math
5
# U ovom modulu se nalaze brojne funkcije kao što su:
7 #
# math.sqrt(broj)
9 # math.log(broj, osnova)
# math.sin(ugao_u_radijanima), math.cos(), ...
11 # math.exp(stepen)
# math.factorial(broj)
13 # i druge...
print "\n-----Matematičke funkcije-----\n"
15 print math.factorial(6)
print math.log(125, 5)
17
# Pseudo slučajni brojevi
19
# Uključujemo modul <<random>>
```



```

21 import random

23 # Funkcija random() vraća pseudo slučajan broj tipa float iz opsega [0.0, 1.0)
print "\n-----Pseudo slučajni brojevi-----\n"
25 print "Pseudo slučajan broj iz opsega [0.0,1.0)\n"
print random.random()

27 # Korisne funkcije:
#
29 # randint(a,b) - vraća pseudo slučajan ceo broj n iz opsega [a,b]
31 # choice(lista) - vraća pseudo slučajan element iz liste
#

```

Zadatak 1.4 Liste.

```

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

8 print "\n-----Lista-----\n"
print lista

10 # Prazna lista
12 prazna = []

14 # Pristupanje elementima liste
print "\n-----Pristupanje elementima liste-----\n"
16 # Indeksiranje elemenata liste
print lista[0]
18 print lista[3][1]
print lista[0:3]
20 # Možemo indeksirati liste unazad, pozicija -1 odgovara poslednjem elementu
print lista[-1]
22 # Ukoliko pokušamo da pristupimo elementu liste
# koji se nalazi na poziciji van opsega interpreter će nam prijaviti gresku
24 # IndexError: list index out of range
# print lista[100]

26 print "\n-----Provera da li se element nalazi u listi-----\n"
28 if 1 in lista:
    print "1 se nalazi u listi\n"

30 print "\n-----Korisne funkcije za rad sa listama-----\n"
32 # Ubacivanje elementa na kraj
print "Ubacivanje elementa na kraj liste\n"
34 lista.append(3.14)
print lista

36 # Ubacivanje elementa na određenu poziciju u listi
38 print "\nUbacivanje elementa na određenu poziciju u listi\n"
# list.insert(pozicija, element)
40 lista.insert(2, "Jana")
print lista

42 #
#
44 # Korisne funkcije:
#
46 # list.remove(x) - izbacuje prvo pojavljivanje elementa x iz liste
# list.count(x) - vraća broj koliko puta se element x nalazi u listi
48 # list.index(x) - vraća indeks prvog pojavljivanja elementa x u listi
# len(lista) - vraća broj elemenata liste
50 # del lista[a:b] - briše elemente liste od pozicije a do b
#
#
52 # Nadovezivanje dve liste
print "\n-----Nadovezivanje dve liste-----\n"
54 lista = lista+["Plava", "Zuta", "Crna"]
print lista

56 #
#
58 # Prolazak kroz listu

```

```

print "\n-----Prolazak kroz listu petljom <<for>>-----\n"
60 for i in lista:
    print i
62
# Poredjenje listi
64 #
# Dve liste se porede tako sto se njihovi elementi porede redom leksikografski
66 print "\n-----Poredjenje listi-----\n"
print "[1,2,3] < [1,2,5]"
68 print [1,2,3] < [1,2,5]
print "\n['abc','abc','abc'] < ['abc', 'ab', 'abcd']"
70 print ['abc','abc','abc'] < ['abc', 'ab', 'abcd']
print "\n['a','b','c'] > ['a', 'b']"
72 print ['a','b','c'] > ['a', 'b']

74
# Koriscenje liste kao stek strukture podataka
76 stek = [9,8,7]
# Operacija push je implementirana funkcijom append
78 stek.append(6)
stek.append(5)
80 print "\n-----Ispisujemo stek-----\n"
print stek
82 # Operacija pop je implementirana funkcijom pop
print "\n-----Ispisujemo element dobijem funkcijom pop-----\n"
84 print stek.pop()
print "\n-----Ispisujemo znanje nakon pozivanja funkcije pop-----\n"
86 print stek

```

Zadatak 1.5 Skup, katalog, uredene n-torke.

```

2 # SKUP
#
# Pravljenje skupa od liste
4 print "\n-----Pravljenje skupa od liste-----\n"
6 lista1 = [4,56,34,2,5,6,4,4,6]
skup = set(lista1)
8
for i in skup:
10     print i
12
# Funkcija <<range>>
14 #
# range(kraj)
16 # range(pocetak, kraj[, korak])
print "\n-----Funkcija <<range>>-----\n"
18 brojevi = range(10)
for i in brojevi:
20     print i
22
# KATALOG
#
24 # Katalog je kolekcija uredjenih parova oblika (kljuc, vrednost)
#
# Notacija: {kljuc:vrednost, kljuc:vrednost, ...}
26 print "\n-----Katalog-----\n"
prazna_mapa = {} # prazna mapa
28
30 mapa1 = {'a' : 3, 'b' : 4, 'c' : 5}
32
print mapa1
34
mapa = {"kljuc1":67.7, 6:"Vrednost 2"}
36
# Pristupanje elementima u mapi
print "\n-----Pristupanje elementima u katalogu-----\n"
38 print mapa['kljuc1']
40
# Prolazak kroz mapu
print "\n-----Prolazak kroz katalog-----\n"
42 for kljuc in mapa:

```

```

    print "{0:s} => {1:s}\n".format(str(kljuc),str(mapa[kljuc]))
44
# Korisne funkcije
46 #
# map.keys() - vraca listu kljuceva iz kataloga
48 # map.values() - vraca listu vrednosti iz kataloga
# map.has_key(kljuc) - vraca True/False u zavisnosti od toga da li se element
50 # sa kljucom kljuc nalazi u katalogu

52 # Uredjene N-TORKE
print "\n-----Torke-----\n"
54 torka = ("Daffy","Duck",11)

56 # Pristupanje elementima u torci
print "\n-----Pristupanje elementima u torci-----\n"
58 print torka[1]

60 print "\n-----Ispisivanje torke-----\n"
print torka

62
# Poredjenje torki
64 #
# Dve torke se porede tako sto se njihovi elementi porede redom leksikografski
66 print "\n-----Poredjenje torki-----\n"
print "(1,2,'a') < (1,2,'b')"
68 print (1,2,'a') < (1,2,'b')
print "\n([1,2,3], 'Bugs', 4) < ([1,1,1], 'Bunny', 6)"
70 print ([1,2,3], 'Bugs', 4) < ([1,1,1], 'Bunny', 6)
# Ukoliko torke ne sadrže elemente istog tipa na istim pozicijama, i dalje ih mozemo
# porediti,
72 # ali poredjenje se vrši na osnovu imena tipa elementa leksikografski
# npr. element tipa List < element tipa String < element tipa Tuple i slicno
74 print "\n(1,2,['a','b']) < (1,2,'ab')"
print (1,2,['a','b']) < (1,2,'ab')
```

1.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.6 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.

[Rešenje 1.6]

Zadatak 1.7 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).

[Rešenje 1.7]

Zadatak 1.8 X-O Napisati program koji implementira igricu X-O sa dva igrača.

[Rešenje 1.8]

1.1.3 Zadaci za vežbu

Zadatak 1.9 Anjc Napisati program koji implementira igricu Anjc 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. <https://en.wikipedia.org/wiki/Blackjack>

Zadatak 1.10 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. https://en.wikipedia.org/wiki/Connect_Four

1.2 Datoteke, niske, JSON format, datum

1.2.1 Uvodni primeri

Zadatak 1.11 Funkcije za rad sa niskama.

```

1 # Niske
2 #
3 # Mozemo ih pisati izmedju jednostrukih i dvostrukih navodnika
4
5 niska1 = 'Ovo je neka niska.'
6 niska2 = "People are strange when you're a stranger ."
7
8 print "\n-----Niske-----\n"
9 print niska1
10 print niska2
11
12 # Karakterima u niski mozemo pristupati koristeći notaciju [] kao kod listi
13 print "\n-----Pristupanje karakterima u niski-----\n"
14 print niska2[4]
15 print niska2[6:10]
16
17 # Duzinu niske racunamo koristeći funkciju len(niska)
18 print "\n-----Duzina niske-----\n"
19 print len(niska1)
20
21 # Funkcija count
22 # niska.count(podniska [, pocetak [, kraj]]) - vraca broj koliko se puta
23 # podniska nalazi u niski (u intervalu od pocetak do kraj)
24 print "\n-----Funkcija <<count>>-----\n"
25 print niska2.count("strange")
26
27 # Funkcija find
28 # niska.find(podniska [, pocetak [, kraj]]) - vraca poziciju prvog pojavljivanja
29 # podniska u niski (u intervalu od pocetak do kraj), -1 ukoliko se podniska ne nalazi
30 # u niski
31 print "\n-----Funkcija <<find>>-----\n"
32 print niska2.find("are")
33
34 # Funkcija join
35 # niska_separator.join([niska1,niska2,niska3,...]) - spaja listu niski separatorom
36 print " ".join(["Olovka", 'pise', 'srcem.'])
37
38 # Korisne funkcije za rad sa niskama:
39 #
40 # niska.isalnum()
41 #     isalpha()
42 #     isdigit()
43 #     islower()

```

```

#         isspace()
46 #         isupper()
# niska.split(separator) - razlaze nisku u listu koristeći separator
48 # niska.replace(stara, nova [, n]) - zamenjuje svako pojavljivanje niske stara
# niskom nova (ukoliko je zadat broj n, onda zamenjuje najviše n pojavljivanja)

```

Zadatak 1.12 Datoteke.

```

1 # Datoteke
#
3 # Datoteku otvaramo koristeći funkciju
#
5 # open(ime_datoteke, mod)
#
7 # mod: "r" -> read, "w" -> write, "a" -> append, "r+" -> read + append
#
9 # Datoteku zatvaramo koristeći funkciju
#
11 # datoteka.close()

13 f = open("dat1.txt", "r")

15 # f.read(n) cita n karaktera iz datoteke
print "\n-----Funkcija <<read>>-----\n"
17 while True:
    c = f.read(2)
19     if c == '':
        break
21     print c

23 # f.readline() cita jednu liniju iz Datoteke
f.close()

25 g = open("dat2.txt", "r")

27 # Liniju po liniju mozemo ucitavati koristeći petlju
29 # tako sto 'iteriramo' kroz Datoteku
print "-----Iteriranje kroz datoteku <<for>> petljom-----\n"
31 for linija in g:
    print linija

33 g.close()
35 # f.readlines() i list(f)
# vraćaju listu linija datoteke
#
37 # f.write(niska) upisuje nisku u datoteku
39 print "-----Upisivanje u datoteku-----\n"
h = open("dat3.txt", "r+")
41 h.write("water\n")

43 print h.readlines()

45 h.close()

```

Zadatak 1.13 Modul datetime.

```

1 # Datumi

3 # Uključujemo klasu datetime iz modula datetime

5 from datetime import datetime

7 # Nov objekat datuma:
#
9 # datetime.datetime(godina, mesec, dan [, sat [, minut [, sekund]])
#
11 # Korisne funkcije:
#
13 # datetime.now() - vraća trenutno vreme odnosno datum
# datetime.strptime(datum_niska, format)
15 # datetime.year, datetime.month, datetime.day, datetime.hour, datetime.minute,
    datetime.second,

```

```
# datetime.strptime(format) - vraća string reprezentaciju objekta datuma na osnovu
#   zadatog formata
17 # datetime.strptime(niska, format) - vraća objekat datetime konstruisan na osnovu
#   niske u zatom formatu
# datetime.time([sat [, minut [, sekund]]) - vraća objekat koji predstavlja vreme
19 # datetime.date(dan, mesec, godina) - vraća objekat datuma
# format:
21 # %A - dan u nedelji (Monday, Tuesday,...)
# %w - dan u nedelji (0, 1, 2,..., 6)
23 # %d - dan (01, 02, 03,...)
# %B - mesec (January, February,...)
25 # %m - mesec (01, 02, ...)
# %Y - godina (1992, 1993,...)
27 # %H - sat (00, 01, ..., 23)
# %M - minut (00, 01, ..., 59)
29 # %S - sekund (00, 01, ..., 59)
#
31
33
35 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()
37 print datetime.now().date()
```

Zadatak 1.14 JSON format.

```
1 # JSON format
#
3 # Funkcije za rad sa JSON formatom se nalaze u modulu json
import json
5
# json.dumps(objekat) vraća string koji sadrži JSON reprezentaciju objekta x
7
9 print "\n-----JSON reprezentacija objekta-----\n"
junak = {"Ime": "Dusko", "Prezime": "Dugousko", "Godine": 11}
print json.dumps(junak)
11
# json.dump(x,f) upisuje string sa JSON reprezentacijom objekta x u datoteku f
13
f = open("dat4.json", "w")
15 json.dump(junak, f)
f.close()
17
# json.load(f) učitava iz datoteke string koji sadrži JSON format objekta i vraća
objekat
19 print "\n-----Učitavanje objekta iz datoteke-----\n"
f = open("dat4.json", "r")
21 x = json.load(f)
print x['Ime']
23 print x['Prezime']
print x['Godine']
25 f.close()
```

1.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.15 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`.

Na primer:

Listing 1.1: `dat.txt`

```
1 Ovo je datoteka dat
```

Listing 1.2: `rezultat.json`

```

1  {
2  'a ': 1, 'ka': 1, 'ot': 1, 'ek': 1,
3  'd ': 2, 'j ': 1, 'da': 2, 'e ': 1,
4  'o ': 1, 'to': 1, 'at': 2, 'je': 1,
5  'Ov': 1, 'te': 1, 'vo': 1
6  }

```

[Rešenje 1.15]

Zadatak 1.16

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

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

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

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

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

[Rešenje 1.16]

1.2.3 Zadaci za vežbu

Zadatak 1.17 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.

Listing 1.3: `ispiti.json`

```

1 [ { 'ime': 'Relacione baze podataka',
2   'datum': '21.09.2016.' },
3   { 'ime': 'Vestacka inteligencija',
4   'datum': '17.06.2017.' },
5   { 'ime': 'Linearna algebra i analiticka geometrija',
6   'datum': '08.02.2017.' } ]

```

Zadatak 1.18 Napisati program koji izdvaja sve jednolinijske i višelinijске 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 višelinijски između `/*` i `*/`.

Listing 1.4: `program.c`

```

#include <stdio.h>
2
// Primer jednolinijskog komentara
4
int main(){
6 /*
   Na ovaj nacin ispisujemo tekst
   na standardni izlaz koristeći jezik C.
8 */
   printf("Hello world!");
10
   // Na ovaj nacin se ispisuje novi red
12   printf("\n");
14 /*
   Ukoliko se funkcija uspesno završila
16   vratamo 0 kao njen rezultat.

```

```
18  */  
    return 0;  
}
```

Listing 1.5: *komentari.json*

```
1 {  
2   'jednolinijski' : ['Primer jednolinijskog komentara',  
3                     'Na ovaj nacin se ispisuje novi red'],  
4   'viselinejski' : ['Na ovaj nacin ispisujemo tekst na standardni  
5                     izlaz koristeći jezik C.',  
6                     'Ukoliko se funkcija uspesno završila  
7                     vracamo 0 kao njen rezultat.'],  
8 }
```

Zadatak 1.19 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.

Listing 1.6: *dat1.txt*

```
2 //netacno  
   same=1;  
4  
   for(i=0;s1[i]!='\0' && s2[i]!='\0';i++) {  
6     if(s1[i]!=s2[i]) {  
       same=0;  
8       break;  
     }  
10  }  
   return same;
```

Listing 1.7: *dat2.txt*

```
1 //tacno  
3  
   for(i=0;s1[i]!='\0' && s2[i]!='\0';i++){  
5     if(s1[i]!=s2[i])  
       return 0;  
7   }  
   return s1[i]==s2[i];
```

Listing 1.8: *razlike.json*

```
1 {  
2   'dat1.txt' : 7,  
3   'dat2.txt' : 4  
4 }
```

1.3 Argumenti komandne linije, sortiranje, obilazak direktorijuma

1.3.1 Uvodni primeri

Zadatak 1.20 Modul `sys` i argumenti komandne linije

```
1 # 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

16 # korisnik moze da zada ime datoteke kao prvi argument komandne linije
18 # u tom slucaju datoteku otvaramo sa f = open(sys.argv[1], "r")
```

Zadatak 1.21 Modul os

```
# Prolazak kroz direktorijume
2
import os
4 # ispisuje oznaku za tekuci direktorijum
print os.getcwd()
6 # ispisuje oznaku za roditeljski direktorijum tekuceg direktorijuma
print os.pardir
8 # ispisuje separator koji koristi za pravljenje putanja
print os.sep

10

12 # funkcija za prosledjenu putanju direktorijuma vraca listu imena svih fajlova u tom
    direktorijumu, . je zamena za putanju tekuceg direktorijuma

14 print os.listdir(".")

16 #
# os.walk() - vraca listu torki (trenutni_direktorijum, poddirektorijumi, datoteke)
18 # os.path.join(putanja, ime) - pravi putanju tako sto nadovezuje na prosledjenu
    putanju zadato ime odvojeno /

20 print "\n-----Prolazak kroz zadati direktorijum-----\n"
for (trenutni_dir, poddirektorijumi, datoteke) in os.walk("."):
22     print trenutni_dir
    for datoteka in datoteke:
24         print os.path.join(trenutni_dir, datoteka)

26 # 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
28 # os.path.isfile(path) - vraca True ako je path putanja regularnog fajla, inace vraca
    False

30 print "\n-----Regularni fajlovi zadataog direktorijuma-----\n"
for ime in os.listdir("."):
32     # ako je regularan fajl u pitanju ispisujemo njegovu apsolutnu putanju
    if os.path.isfile(os.path.join(".", ime)):
34         print os.path.abspath(os.path.join(".", ime))
```

Zadatak 1.22 Sortiranje

```
# Sortiranje
#
3 # sorted(kolekcija [, poredi [, kljuc [, obrni]]) - vraca sortiranu kolekciju
#
5 # kolekcija - kolekcija koju zelimo da sortiramo
# poredi - funkcija poredjenja
7 # kljuc - funkcija koja vraca kljuc po kome se poredi
# obrni - True/False (opadajuce/rastuce)
9 #
# za poziv sorted(kolekcija) koristi se funkcija cmp za poredjenje
11 # cmp(x, y) -> integer
```

```

# vraca negativnu vrednost za x<y, 0 za x==y, pozitivnu vrednost za x>y
13 # ako su x i y niske, cmp ih leksikografski poredi
#
15
import json
17 import math

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

23 # u sledecem primeru je neophodno da definisemo svoje funkcije za poredjenje i
# vracanje kljucja jer je kolekcija lista recnika i za to cmp nema definisano
# ponasanje
tacke = [{"teme": "A", "koordinata": [10.0, 1.1]}, {"teme": "B", "koordinata": [1.0,
15.0]}, {"teme": "C", "koordinata": [-1.0, 5.0]}]
25

27 # funkcija koja tacke x i y poredi po njihovoj udaljenosti od koordinatnog pocetka
def poredi(x,y):
29     if (x[0]*x[0] + x[1]*x[1]) > (y[0]*y[0] + y[1]*y[1]):
        return 1
31     else:
        return -1
33 # 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
35 def kljuc(x):
    return x["koordinata"]
37

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

```

1.3.2 Zadaci za samostalni rad sa rešenjima

Zadatak 1.23 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.

[Rešenje 1.23]

Zadatak 1.24 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`. Na primer:

Listing 1.9: `rezultat.txt`

```

1 {
2   'txt' : 14,
3   'py'  : 12,
4   'c'   : 10
5 }
6

```

[Rešenje 1.24]

Zadatak 1.25 U datoteci `radnici.json` nalaze se podaci o radnom vremenu zaposlenih u preduzeću u sledecem formatu:

```

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

```

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.

[Rešenje 1.25]

Zadatak 1.26 Napisati program koji učitava ime datoteke sa standardnog ulaza i na standardni izlaz ispisuje putanje do svih direktorijuma u kojima se nalazi ta datoteka.

[Rešenje 1.26]

1.3.3 Zadaci za vežbu

Zadatak 1.27 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.

Zadatak 1.28 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`.

1.4 Rešenja

Rešenje 1.6 Pogodi broj

```

1 # Pogodi broj
3 import random
5 print "----- IGRA: Pogodi broj -----\n"
7 zamisljen_broj = random.randint(0,100)
9 ime = raw_input("Unesite Vase ime: ")
11 print "Zdravo {0:s}. :) \nZamisljio sam neki broj od 1 do 100. Da li mozes da pogodis
    koji je to broj?".format(ime)
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 zamisljio je MANJI od {0:d}.".format(broj)
21     else:
            print "Broj koji sam zamisljio je VECI od {0:d}.".format(broj)
23 print "BRAVO!!! Pogodio si! Zamisljio sam {0:d}. Bilo je lepo igrati se sa tobom. :)".
    format(zamisljen_broj)

```

Rešenje 1.7 Aproksimacija broja PI metodom Monte Karlo

```
# Aproksimacija broja PI metodom Monte Karlo
2
4 import random
6 print "Izracunavanje broja PI metodom Monte Karlo \n"
N = int(raw_input("Unesite broj iteracija: "))
8 # Broj tacaka koje se nalaze u krugu
A = 0
10 # Broj tacaka koje se nalaze u kvadratu
B = 0
12
13 i = N
14 while i >= 0:
    tacka = (random.random(), random.random())
16     print "Tacka: "
    print tacka
18     # Ukoliko se tacka nalazi u krugu, povecavamo broj tacaka u krugu
    if ((float(tacka[0])-0.5)**2 + (float(tacka[1])-0.5)**2) < (0.5**2):
20         A = A + 1
        B = B + 1
22         i = i - 1
24
25 print "Broj PI aproksimiran metodom Monte Karlo: "
print 4.0*A/B
```

Rešenje 1.8 X-O

```
# X-O
#
3 # - | 0 | X
# ---
5 # X | - | -
# ---
7 # - | X | 0
9
10 import random
11
12 def ispisi_tablu(tabla):
    print "\n TABLA \n"
13     print " 1 2 3 "
    print " --- "
15     indeks = 1
    for i in tabla:
17         print indeks,"|",i[0],"|",i[1],"|",i[2],"|"
        print " --- "
19         indeks = indeks + 1
    print "\n"
21
22 def pobedio(tabla):
23     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
25     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]))):
27         return True
    return False
29
30 def ucitaj_koordinate(ime):
31     while True:
        print "{0:s} unesite koordinate polja koje zelite da popunite u posebnim
linijama:\n".format(ime)
33         x = int(raw_input("Unesite vrstu: "))
        y = int(raw_input("Unesite kolonu: "))
35         if 1<=x<=3 and 1<=y<=3:
            return x-1,y-1
37         else:
            "Morate uneti brojeve 1,2 ili 3\n"
39
40 def korak(igrac):
```

```

41     while True:
42         x,y = učitaj_koordinate(igrac[0])
43         if tabla[x][y] == "-":
44             tabla[x][y] = igrac[1]
45             ispisi_tablu(tabla)
46             break
47         else:
48             print tabla[x][y]
49             print "Uneto polje je popunjeno!\n"
51 print "IGRA: X-O pocinje\n"
53 ime1 = raw_input("Unesite ime prvog igraca: ")
54 print "Zdravo {0:s}!\n".format(ime1)
55 ime2 = raw_input("Unesite ime drugog igraca: ")
56 print "Zdravo {0:s}!\n".format(ime2)
57
58 indikator = random.randint(1,2)
59 if indikator == 1:
60     prvi_igrac = (ime1, "X")
61     drugi_igrac = (ime2, "O")
62 else:
63     prvi_igrac = (ime2, "X")
64     drugi_igrac = (ime1, "O")
65
66 print "Igrac {0:s} igra prvi. \n".format(prvi_igrac)
67 print "X : {0:s}\n".format(prvi_igrac)
68 print "O : {0:s}\n".format(drugi_igrac)
69
70 tabla = [['-', '-', '-'], ['-', '-', '-'], ['-', '-', '-']]
71
72 print "Zapocnimo igru \n"
73
74 ispisi_tablu(tabla)
75
76 na_redu = 0
77 iteracija = 0
78 igraci = [prvi_igrac, drugi_igrac]
79 while iteracija < 9:
80     korak(igraci[na_redu])
81     if pobedio(tabla) == True:
82         print "BRAVO!!!!!! {0:s} je pobedio!\n".format(igraci[na_redu][0])
83         break
84     na_redu = (na_redu+1)%2
85     iteracija = iteracija + 1
87
88 if iteracija == 9:
89     print "NERESENO! Pokusajte ponovo.\n"

```

Rešenje 1.15

```

# dat.txt:
2 # Ovo je datoteka dat
#
4 # rezultat.json:
#
6 # {"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}
8 import json
10 ime_datoteke = raw_input("Unesite ime datoteke: ")
11 n = int(raw_input("Unesite broj n: "))
12
13 # Otvaramo datoteku i citamo njen sadrzaj
14 f = open(ime_datoteke, "r")
15 sadrzaj = f.read()
16 f.close()
17
18 recnik = {}
19 i = 0
20 # Prolazimo kroz sadrzaj i uzimamo jedan po jedan n-gram

```

```
22 while i < len(sadrzaj) - n:
    ngram = sadrzaj[i : i+n]
    # Ukoliko se n-gram vec nalazi u recniku,
    # povecavamo mu broj pojavljivanja
    24 if ngram in recnik:
        26 recnik[ngram] = recnik[ngram]+1
        # Dodajemo n-gram u recnik i postavljamo mu broj na 1
    28 else:
        recnik[ngram] = 1
    30 i = i + 1

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

Rešenje 1.16

```
1 import json

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

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

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

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

19 maxi_racun = 0
21 idea_racun = 0
    shopngo_racun = 0
23 # Za svako voce u korpi dodajemo njegovu cenu u svaki racun posebno
    for voce in korpa:
25         ime = voce['ime']
            maxi_racun = maxi_racun + korpa[ime]['kolicina']*maxi_cene[ime]['cena']
27         idea_racun = idea_racun + korpa[ime]['kolicina']*idea_cene[ime]['cena']
            shopngo_racun = shopngo_racun + korpa[ime]['kolicina']*shopngo_cene[ime]['cena']
29
31 print "Maxi: " + str(maxi_racun) + " dinara"
    print "Idea: " + str(idea_racun) + " dinara"
    print "Shopngo: " + str(shopngo_racun) + " dinara"
```

Rešenje 1.23

```
1 import os

3 dat_u_duze = 0
5 dat_u_sire = 0

7 # Funkcija koja obilazi datoteku i vraca 1 ukoliko datoteka pripada skupu duze
    # odnosno 0 ukoliko datoteka pripada skupu sire
9 def obilazak(ime_datoteke):
    br_linija = 0
11    najduza_linija = 0
        f = open(ime_datoteke, "r")
13    for linija in f:
            br_linija = br_linija + 1
15            if len(linija) > najduza_linija:
                najduza_linija = len(linija)
17    f.close()
    if br_linija > najduza_linija:
```

```

19         return 1
20     else:
21         return 0
22
23 ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")
24
25 for (tren_dir, pod_dir, datoteke ) in os.walk(ime_direktorijuma):
26     for dat in datoteke:
27         if obilazak(os.path.join(tren_dir, dat)) == 0:
28             dat_u_sire += 1
29         else:
30             dat_u_duze += 1
31
32 print "Kardinalnost skupa duze: kardinalnost skupa sire"
33 print str(dat_u_duze)+" "+str(dat_u_sire)

```

Rešenje 1.24

```

1 import os
2 import json
3
4 ime_direktorijuma = raw_input("Unesite putanju do direktorijuma: ")
5
6 ekstenzije = {}
7
8 for (tren_dir, pod_dir, datoteke ) in os.walk(ime_direktorijuma):
9     for dat in datoteke:
10        pozicija = dat.find(".")
11        # Ukoliko datoteka ima ekstenziju, pretpostavljamo da su datoteke imenovane
12        # tako da posle . ide ekstenzija u ispravnom obliku
13        if pozicija >= 0:
14            # Ukoliko ekstenzija postoji u mapi, povecavamo njen broj
15            if dat[pozicija:] in ekstenzije:
16                ekstenzije[dat[pozicija:]] += 1
17            else:
18                # Dodajemo novu ekstentiju u mapu i postavljamo njen broj na 1
19                ekstenzije[dat[pozicija:]] = 1
20
21 f = open("rezultat.json","w")
22 json.dump(ekstenzije, f)
23 f.close()

```

Rešenje 1.25

```

1 import json, os
2 from datetime import datetime
3
4 f = open("radnici.json", "r")
5 radnici = json.load(f)
6 f.close()
7
8 opcija = raw_input("Unesite opciju koju zelite (d - dostupni radnici, o - radnici na
9     odmoru): \n")
10
11 if opcija != "d" and opcija != "o":
12     print "Uneta opcija nije podrzana."
13     exit()
14
15 tren_dat = datetime.now()
16
17 # funkcija datetime.strptime(string, format) pravi objekat tipa datetime na osnovu
18 # zadatih podataka u stringu i odgovarajuceg formata, na primer ako je datum
19 # zapisan kao "21.08.2016" odgovarajuci format je "%d.%m.%Y." pa se funkcija poziva
20 # sa datetime.strptime("21.08.2016", "%d.%m.%Y.")
21
22 for radnik in radnici:
23     kraj_odmora = datetime.strptime(radnik['odmor'][1], "%d.%m.%Y.").date()
24     pocetak_odmora = datetime.strptime(radnik['odmor'][0], "%d.%m.%Y.").date()
25     kraj_rad_vrem = datetime.strptime(radnik['radno_vreme'][1], "%H:%M").time()

```

```
23 pocetak_rad_vrem = datetime.strptime(radnik['radno_vreme'][0], "%H:%M").time()
if opcija == "o":
    # Ukoliko je radnik trenutno na odmoru ispisujemo ga
25     if pocetak_odmora < tren_dat.date() < kraj_odmora:
        print radnik["ime"]
27     else:
        # Ukoliko je radnik trenutno dostupan i nije na odmoru, ispisujemo ga
29     if not (pocetak_odmora < tren_dat.date() < kraj_odmora) and pocetak_rad_vrem
        < tren_dat.time() < kraj_rad_vrem:
            print radnik["ime"]
```

Rešenje 1.26

```
import os
2
ime_datoteke = raw_input("Unesite ime datoteke: ")
4
# pretražujemo ceo fajl sistem, odnosno pretragu krecemo od root direktorijuma /
# imajte u vidu da ce vreme izvršavanja ovog programa biti veliko posto se pretražuje
6 # ceo fajl sistem, mozete ga prekinuti u svakom trenutku sa CTRL+C
for (tren_dir, pod_dir, datoteke ) in os.walk("/"):
8     # objekat datoteke predstavlja listu imena datoteka iz direktorijuma
    # ta imena poredimo sa zadatim
10     for dat in datoteke:
        # ako smo naisli na trazenu datoteku, pravimo odgovarajucu putanju
12     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 može instalirati pomoću 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 Modul constraint, osnovne funkcije

```
1 # Programiranje ogranicenja
3 # Uključujemo modul za rad sa ogranicenjima
import constraint
5
# Definiramo problem
7 problem = constraint.Problem()
# Dodajemo promenljive
9 #
# problem.addVariable(ime_promenljive, domen_lista)
11 # problem.addVariables(lista_imena_promenljivih, domen_lista)
problem.addVariable('x',[1,2,3])
13 problem.addVariable('y',['a','b','c'])
# Ispisujemo resenja
15 # print problem.getSolutions()

17 problem.addVariable('z',[0.1,0.2,0.3])
# Dodajemo ogranicenja
19 #
# problem.addConstraint(ogranicenje [, redosled_promenljivih])
21 #
# ogranicenje može biti:
23 # constraint.AllDifferentConstraint() - različite vrednosti svih promenljivih
# constraint.AllEqualConstraint() - iste vrednosti svih promenljivih
25 # constraint.MaxSumConstraint(s [,tezine]) - suma vrednosti promenljivih (pomnožena
sa tezinama) ne prelazi s
# constraint.MinSumConstraint(s [,tezine]) - suma vrednosti promenljivih (pomnožena
sa tezinama) nije manja od s
27 # constraint.ExactSumConstraint(s [,tezine]) - suma vrednosti promenljivih (
pomnožena sa tezinama) je s
# constraint.InSetConstraint(skup) - vrednosti promenljivih se nalaze u skupu skup
```

```
29 # constraint.NotInSetConstraint(skup) - vrednosti promenljivih se ne nalaze u skupu
    skup
    # constraint.SomeInSetConstraint(skup) - vrednosti nekih promenljivih se nalaze u
    skupu skup
31 # constraint.SomeNotInSetConstraint(skup) - vrednosti nekih promenljivih se ne
    nalaze u skupu skup
    #
33 # redosled_promenljivih predstavlja listu promenljivih
    # i zadaje se zbog definisanja tacnog redosleda
35 # ogranicenja koja se primenjuju na promenljive
    #
37 # Mozemo napraviti i svoju funkciju ogranicenja
    def ogranicenje(x,z):
39     if x / 10.0 == z:
        return True
41
    # Prosledjujemo funkciju ogranicenja i redosled promenljivih koji treba da odgovara
    redosledu argumenata funkcije ogranicenja
43 problem.addConstraint(ogranicenje,['x','z'])
    resenja = problem.getSolutions()
45 print "\n-----Resenja-----\n"
    for resenje in resenja:
47     print resenje
```

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.

[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.

[Rešenje 2.3]

Zadatak 2.4 Napisati program koji reda 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 2.4]

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

[Rešenje 2.5]

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

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

[Rešenje 2.6]

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

[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

```

[Rešenje 2.8]

Zadatak 2.9 Pekara *Kiftica* 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 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 poredane 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 ($A+B+C = D+E+F+G = \dots = Q+R+S = 38$, $A+D+H = B+E+I+M = \dots = L+P+S = 38$, $C+G+L = B+F+K+P = \dots = H+M+Q = 38$).

[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 način kompanija treba da obuči svoje zaposlene kako bi ostvarila maksimalnu dobit.

[Rešenje 2.11]

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

[Rešenje 2.12]

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

[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.

[Rešenje 2.14]

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
```

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
```

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

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 $n - 1$. $S = (x_0, x_1, \dots, x_{n-1})$ je magična sekvenca ukoliko postoji x_i pojavljivanja broja i za $i = 0, 1, \dots, n - 1$.

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?

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?

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 prodanih 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 obogatilo?

2.2 Rešenja

Rešenje 2.2

```

1  import constraint
3
5  problem = constraint.Problem()
6  # Definisemo promenljive i njihove vrednosti
7  problem.addVariable('A',range(1,10))
8  problem.addVariable('B',range(10))
9  problem.addVariable('C',range(10))
10 # Dodajemo ogranicenje da su vrednosti svih promenljivih razlicite
11 problem.addConstraint(constraint.AllDifferentConstraint())
12 resenja = problem.getSolutions()
13 # Znamo da minimalni kolicnik mora biti manji od 999
14 min_kolicnik = 999
15 min_resenje = {}
16 for resenje in resenja:
17     a = resenje['A']
18     b = resenje['B']
19     c = resenje['C']
20     kolicnik = (float)(a*100 + b*10 + c) / (a+b+c)
21     if kolicnik < min_kolicnik:
22         min_kolicnik = kolicnik
23         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 promenljivu kreće od 1
problem.addVariable("1 din",range(0,51))
7 problem.addVariable("2 din",range(0,26))
problem.addVariable("5 din",range(0,11))
9 problem.addVariable("10 din",range(0,6))
problem.addVariable("20 din",range(0,3))
11
# Problem koji je uocen pri ispisu resenja je sledeci, redosled u kom ce biti dodate
  promenljive problemu ne mora uvek da odgovara redosledu kojim smo mi definisali
  promenljive, u konkretnom primeru (videti oblik u kom ispisuje resenje),
  promenljive ce se dodati u sledecem redosledu: '1 din', '2 din', '10 din', '20 din',
  '5 din' (nacin na koji se kljucevi organizuju u rečniku nije striktno
  definisan, primetimo da niske nisu sortirane)
13 # posledica je da postavljanje ograničenja
# problem.addConstraint(constraint.ExactSumConstraint(50,[1,2,5,10,20]))
15 # nece ispravno dodeliti tezine, na primer, tezinu 5 dodeli promenljivoj '10 din'
  umesto '5 din' kako bismo ocekivali
17 # I nacin da se resi ovaj problem je da redosled promenljivih koji odgovara redosledu
  tezina za ExactSumConstraint prosledimo kao dodatni argument za funkciju
  addConstraint
19 problem.addConstraint(constraint.ExactSumConstraint(50,[1,2,5,10,20]), ["1 din", "2
  din", "5 din", "10 din", "20 din"])
21 # II nacin je da definisemo svoju funkciju koja predstavlja ograničenje, samo ce sada
  solver nesto sporije da radi posto ugradjene funkcije imaju optimizovanu
  pretragu i brze dolaze do resenja
#
23 #def o(a, b, c, d, e):
# if a + 2*b + 5*c + 10*d + 20*e == 50:
25 #     return True
#
27 #problem.addConstraint(o, ["1 din", "2 din", "5 din", "10 din", "20 din"])
#
29 resenja = problem.getSolutions()
31 for r in resenja:
  print r
33 # 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
2 # 3 5 7
# 8 1 6
4 #
6 import constraint
8 def o(x,y,z):
  if x+y+z == 15:
10     return True
12 problem = constraint.Problem()
# Promenljive:
14 # a b c
# d e f
16 # g h i
problem.addVariables("abcdefghi", range(1,10))
```

```

18 problem.addConstraint(constraint.AllDifferentConstraint())
# Dodajemo ogranicenja za svaku vrstu
20 problem.addConstraint(o,"abc")
problem.addConstraint(o,"def")
22 problem.addConstraint(o,"ghi")
# Dodajemo ogranicenja za svaku kolonu
24 problem.addConstraint(o,"adg")
problem.addConstraint(o,"beh")
26 problem.addConstraint(o,"cfi")
#Dodajemo ogranicenja za dijagonale
28 problem.addConstraint(o,"aei")
problem.addConstraint(o,"ceg")
30
resenja = problem.getSolutions()
32 for r in resenja:
    print "-----"
34     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'])
36     print "| {0:d} {1:d} {2:d} |".format(r['g'],r['h'],r['i'])
    print "-----"

```

Rešenje 2.5

```

1 # X,Y,Z
#
3 # X >= Z
# X*2 + X*Y + Z <= 34
5 #
# X <- {1,2,3,...90}
7 # Y <- {2,4,6,...60}
# Z <- {1,4,9,16,...100}
9 #
#
11 import constraint
13
14 problem = constraint.Problem()
15
# Dodajemo promenljivu X i definisemo njen domen
17 problem.addVariable('X', range(1,91))
18
# Dodajemo promenljivu Y i definisemo njen domen
19 problem.addVariable('Y', range(2,61,2))
20
21 domenZ = [];
23 for i in range(1,11):
    domenZ.append(i*i)
25
# Dodajemo promenljivu Z i definisemo njen domen
27 problem.addVariable('Z', domenZ)
28
29 def o1(x,z):
    if x >= z:
31         return True
32
33 def o2(x,y,z):
    if x*2 + x*y + z <= 34:
35         return True;
36
37 # Dodajemo ogranicenja
problem.addConstraint(o1, 'XZ')
39 problem.addConstraint(o2, 'XYZ')
40
41 resenja = problem.getSolutions()
42
43 for r in resenja:
    print "-----"
45     print "X = {0:d} , Y = {1:d} , Z = {2:d}".format(r['X'],r['Y'],r['Z'])
    print "-----"

```

Rešenje 2.6

```

2 # TWO
3 # +TWO
4 # -----
5 # FOUR
6 #
7
8 import constraint
9
10 problem = constraint.Problem()
11 # Definiramo promenljive i njihove vrednosti
12 problem.addVariables("TF", range(1,10))
13 problem.addVariables("WOUR", range(10))
14
15 # Definiramo ogranicenje za cifre
16 def o(t, w, o, f, u, r):
17     if 2*(t*100 + w*10 + o) == f*1000 + o*100 + u*10 + r:
18         return True
19
20 # Dodajemo ogranicenja za cifre
21 problem.addConstraint(o, "TWOFUR")
22 # Dodajemo ogranicenje da su sve cifre razlicite
23 problem.addConstraint(constraint.AllDifferentConstraint())
24
25 resenja = problem.getSolutions()
26
27 for r in resenja:
28     print "-----"
29     print "  "+str(r['T'])+str(r['W'])+str(r['O'])
30     print " "+str(r['T'])+str(r['W'])+str(r['O'])
31     print "="+str(r['F'])+str(r['O'])+str(r['U'])+str(r['R'])

```

Rešenje 2.7

```

2 # Dati sistem nejednacina nema resenje, tj. metog getSolutions() vraca praznu listu
3 # Ukoliko se za promenljivu W domen promeni na {1,...,100} sistem ce imati resenje
4 import constraint
5
6 problem = constraint.Problem()
7
8 # Dodajemo promenljivu X i definisemo njen domen
9 problem.addVariable('X', range(1,11))
10
11 # Dodajemo promenljivu Y i definisemo njen domen
12 problem.addVariable('Y', range(1,52,2))
13
14 domenZ = []
15 domenW = []
16
17 for i in range(1,11):
18     domenZ.append(i*10)
19     domenW.append(i**3)
20
21 # Dodajemo promenljivu Z i definisemo njen domen
22 problem.addVariable('Z', domenZ)
23
24 # Dodajemo promenljivu W i definisemo njen domen
25 problem.addVariable('W', domenW)
26
27 # Za ovako definisan domen za promenljivu W sistem ce imati resenje
28 # problem.addVariable('W', range(1,101))
29
30 def o1(x,w):
31     if x >= 2*w:
32         return True
33
34 def o2(y,z):
35     if 3 + y <= z:
36         return True

```



```

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

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

46
resenja = problem.getSolutions()
48 # Proveravamo da li postoji resenje za sistem nejednacina
if resenja==[]:
50     print "Sistem nema resenje."
else:
52     for r in resenja:
        print "-----"
54         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 #      1   3
#      2   4
3 #      5
#      6   8
5 #      7   9
#

7
import constraint
9
# Definiseмо ogranicenje za jednu dijagonalu
11 def o(a,b,c,d,e):
    if a<b<c<d<e and a+b+c+d+e==25:
13         return True

15 problem = constraint.Problem()
# Definiseмо promenljive za svaku poziciju
17 problem.addVariables('abcdeABCDE',range(1,10))
# Dodajemo ogranicenja za obe dijagonale
19 problem.addConstraint(o,'abcde')
problem.addConstraint(o,'ABCDE')
21 # Dodajemo ogranicenje da su vrednosti svih promenljivih razlicite
problem.addConstraint(constraint.AllDifferentConstraint())

23
resenja = problem.getSolutions()
25 for r in resenja:
    print "-----"
27     print "{0:d}   {1:d}".format(r['a'],r['A'])
    print " {0:d} {1:d} ".format(r['b'],r['B'])
29     print "  {0:d} ".format(r['c'])
    print " {0:d} {1:d} ".format(r['D'],r['d'])
31     print "{0:d}   {1:d}".format(r['E'],r['e'])
    print "-----"

```

Rešenje 2.9

```

#
2 # Potrebno je napraviti H komada hleba i K komada kifli
#
4 # Zarada iznosi:
# - 7din/hleb, tj. zarada za H komada hleba bice 7*H
6 # - 9din/kifla tj. zarada za K komada kifli bice 9*K
#
8 # Ukupna zarada iznosi:
# 7*H + 9*K - funkcija koju treba maksimizovati
10 #
# Ogranichenja vremena:
12 # - vreme potrebno za mesenje jednog hleba je 10min,
#   tj. za mesenje H komada hleba potrebno je 10*H minuta

```

2 Programiranje ograničenja - Python

```
14 # - vreme potrebno za mesenje jedne kifle je 12min,  
15 #   tj. za mesenje K komada kifli potrebno je 12*K minuta  
16 #  
17 # Ukupno vreme koje je na raspolaganju iznosi 20h, tako da je:  
18 #  $10*H + 12*K \leq 1200$   
19 #  
20 # Ogranicenje materijala:  
21 # - za jedan hleb potrebno je 300g brasna, a za H komada hleba potrebno je H*300  
22 #   grama  
23 # - za jednu kifli potrebno je 120g brasna, a za K komada kifli potrebno je K*120  
24 #   grama  
25 #  
26 # Ukupno, na raspolaganju je 20kg brasna, tako da je:  
27 #  $300*H + 120*K \leq 20000$   
28 #  
29 # Broj kifli i hleba je najmanje 0, tako da:  
30 #  $H \geq 0$   
31 #  $K \geq 0$   
32 #  
33 # S obzirom na to da imamo 20kg brasna na raspolaganju, mozemo napraviti:  
34 # - najvise 20000/120 kifli  
35 # - najvise 20000/300 hleba  
36 #  
37 #  $H \leq 20000/120 \sim 167$   
38 #  $K \leq 20000/300 \sim 67$   
39 #  
40 # S obzirom na to da imamo 20h na raspolaganju, mozemo napraviti:  
41 # - najvise 1200/12 kifli  
42 # - najvise 1200/10 hleba  
43 #  
44 #  $H \leq 1200/10 = 120$   
45 #  $K \leq 1200/12 = 100$   
46 #  
47 # najoptimalnije je za gornju granicu domena postaviti minimum od dobijenih vrednosti  
48 #   , tj. sve ukupno  $H \leq 120$ ,  $K \leq 67$   
49  
50 import constraint  
51  
52 problem = constraint.Problem()  
53  
54 # Dodajemo promenljivu H i definisemo njen domen  
55 problem.addVariable('H', range(0,121))  
56  
57 # Dodajemo promenljivu K i definisemo njen domen  
58 problem.addVariable('K', range(0,68))  
59  
60 def ogranicenje_vremena(h,k):  
61     if 10*h + 12*k <= 1200:  
62         return True  
63  
64 def ogranicenje_materijala(h,k):  
65     if 300*h + 120*k <= 20000:  
66         return True;  
67  
68 # Dodajemo ogranicenja vremena i matrijala  
69 problem.addConstraint(ogranicenje_vremena, 'HK')  
70 problem.addConstraint(ogranicenje_materijala, 'HK')  
71  
72 resenja = problem.getSolutions()  
73  
74 # Pronalazimo maksimalnu vrednost funkcije cilja  
75 max_H = 0  
76 max_K = 0  
77  
78 for r in resenja:  
79     if 7*r['H'] + 9*r['K'] > 7*max_H + 9*max_K:  
80         max_H = r['H']  
81         max_K = r['K']  
82  
83 print "-----"  
84 print "Maksimalna zarada je {0:d}, komada hleba je {1:d}, a komada kifli {2:d}".  
85     format(7*max_H + 9*max_K, max_H, max_K)  
86 print "-----"
```

Rešenje 2.10

```

1 import constraint
3
4 def o1(x,y,z):
5     if x+y+z == 38:
6         return True
7 def o2(x,y,z,w):
8     if x+y+z+w == 38:
9         return True
10 def o3(x,y,z,w,h):
11     if x+y+z+w+h == 38:
12         return True
13
14 problem = constraint.Problem()
15 problem.addVariables("ABCDEFGHJKLMNOPQRST", range(1,38))
16 problem.addConstraint(constraint.AllDifferentConstraint())
17 # Dodajemo ogranicenja za svaku horizontalnu liniju
18 # A,B,C
19 # D,E,F,G
20 #H,I,J,K,L
21 # M,N,O,P
22 # Q,R,S
23 problem.addConstraint(o1,"ABC")
24 problem.addConstraint(o2,"DEFG")
25 problem.addConstraint(o3,"HIJKL")
26 problem.addConstraint(o2,"MNOP")
27 problem.addConstraint(o1,"QRS")
28
29 # Dodajemo ogranicenja za svaku od glavnih dijagonala
30 # A,B,C
31 # D,E,F,G
32 #H,I,J,K,L
33 # M,N,O,P
34 # Q,R,S
35
36 problem.addConstraint(o1,"HMQ")
37 problem.addConstraint(o2,"DINR")
38 problem.addConstraint(o3,"AEJOS")
39 problem.addConstraint(o2,"BFKP")
40 problem.addConstraint(o1,"CGL")
41
42 # Dodajemo ogranicenja za svaku od sporednih dijagonala
43 # A,B,C
44 # D,E,F,G
45 #H,I,J,K,L
46 # M,N,O,P
47 # Q,R,S
48
49 problem.addConstraint(o1,"ADH")
50 problem.addConstraint(o2,"BEIM")
51 problem.addConstraint(o3,"CFJNQ")
52 problem.addConstraint(o2,"GKOR")
53 problem.addConstraint(o1,"LPS")
54
55 resenja = problem.getSolutions()
56 for r in resenja:
57     print " -----"
58     print " {0:d},{1:d},{2:d}".format(r['A'],r['B'],r['C'])
59     print " {0:d},{1:d},{2:d},{3:d}".format(r['D'],r['E'],r['F'],r['G'])
60     print " {0:d},{1:d},{2:d},{3:d},{4:d}".format(r['H'],r['I'],r['J'],r['K'],r['L'])
61     print " {0:d},{1:d},{2:d},{3:d}".format(r['M'],r['N'],r['O'],r['P'])
62     print " {0:d},{1:d},{2:d}".format(r['Q'],r['R'],r['S'])
63     print " -----"

```

Rešenje 2.11

```

1 import constraint
3
4 problem = constraint.Problem()
5 # Kompanija ima 250 zaposlenih radnika

```

```

5 # za sve njih organizuje dodatnu obuku
# ako je E promenljiva za Elixir, a D za Dart
7 # mora da vaziti E<=250, D<=250 i E + D = 250
#
9 # Dodajemo promenljivu E i definisemo njen domen
problem.addVariable('E', range(0,251))
11
13 # Dodajemo promenljivu D i definisemo njen domen
problem.addVariable('D', range(0,251))
15
17 def ukupno_radnika(e,d):
    if e+d == 250:
        return True
19
21 def ogranicenje_projekat_sati(e,d):
    if 150*e + 170*d <= 51200:
        return True
23
25 def ogranicenje_sredstava(e,d):
    if 100*e + 105*d <= 26000:
        return True;
27
29 # Dodajemo ogranicenja za broj projekat/sati i ukupna sredstva
# na raspolaganju kao i za broj radnika u firmi
31 problem.addConstraint(ogranicenje_projekat_sati, 'ED')
problem.addConstraint(ogranicenje_sredstava, 'ED')
problem.addConstraint(ukupno_radnika, 'ED')
33
35 resenja = problem.getSolutions()
37
39 # Pronalazimo maksimalnu vrednost funkcije cilja
max_E = 0
max_D = 0
# Od ostvarene dobiti preko broja projekat/sati oduzimamo gubitak za placanje kurseva
radnicima
41 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']
43
45 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

```

1 # Jedan od mogucih rasporeda
# sva ostala resenja su permutacije
3 # takve da je u svakoj vrsti i koloni samo jedan top
#
5 # 8 T - - - - -
# 7 - T - - - - -
7 # 6 - - T - - - -
# 5 - - - T - - -
9 # 4 - - - - T - -
# 3 - - - - - T -
11 # 2 - - - - - T -
# 1 - - - - - - T
13 # 1 2 3 4 5 6 7 8
#
15
17 import constraint
19
21 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
23 problem.addConstraint(constraint.AllDifferentConstraint())
resenja = problem.getSolutions()

```

```

25 # Broj svih mogućih permutacija je 8! = 40320
26 # Za prikaz svih najbolje pozvati program sa preusmerenjem izlaznih podataka
27 # python 2_12.py > izlaz.txt
28 print "Broj rešenja je: {0:d}".format(len(rešenja))
29
30 for r in rešenja:
31     print "-----"
32     for i in "12345678":
33         for j in range(1,9):
34             if r[i] == j:
35                 print "T",
36             else:
37                 print "-",
38         print ""
39     print "-----"

```

Rešenje 2.13

```

1 # 8 D - - - - -
2 # 7 - - - - D - - -
3 # 6 - D - - - - -
4 # 5 - - - - - D - -
5 # 4 - - D - - - - -
6 # 3 - - - - - D -
7 # 2 - - - - D - - -
8 # 1 - - - - - D
9 # 1 2 3 4 5 6 7 8
10 #
11
12 import constraint
13 import math
14
15 problem = constraint.Problem()
16 # Dodajemo promenljive za svaku kolonu i njihove vrednosti 1-8
17 problem.addVariables("12345678", range(1,9))
18 # Dodajemo ogranicenje da se dame ne napadaju medjusobno po svakoj vrsti
19 problem.addConstraint(constraint.AllDifferentConstraint())
20
21 for k1 in range(1,9):
22     for k2 in range(1,9):
23         if k1 < k2:
24             # Definiseмо funkciju ogranicenja za dijagonale
25             def o(vrsta1, vrsta2, kolona1=k1, kolona2=k2):
26                 if math.fabs(vrsta1 - vrsta2) != math.fabs(kolona1 - kolona2):
27                     return True
28             problem.addConstraint(o, [str(k1),str(k2)])
29
30 rešenja = problem.getSolutions()
31 # Za prikaz svih najbolje pozvati program sa preusmerenjem izlaznih podataka
32 # python 2_13.py > izlaz.txt
33 print "Broj rešenja je: {0:d}".format(len(rešenja))
34 for r in rešenja:
35     print "-----"
36     for i in "12345678":
37         for j in range(1,9):
38             if r[i] == j:
39                 print "D",
40             else:
41                 print "-",
42         print ""
43     print "-----"

```

Rešenje 2.14

```

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

```

```

7 # 3 - - - - -
9 # 2 - - - - -
9 # 1 - - - - -
# 1 2 3 4 5 6 7 8 9
11 #
13 import constraint
13 import json
15
15 problem = constraint.Problem()
17
17 # Dodajemo promenljive za svaki red i njihove vrednosti 1-9
19 for i in range(1, 10):
19     problem.addVariables(range(i * 10 + 1, i * 10 + 10), range(1, 10))
21
21 # Dodajemo ogranicenja da se u svakoj vrsti nalaze razlicite vrednosti
23 for i in range(1, 10):
23     problem.addConstraint(constraint.AllDifferentConstraint(), range(i * 10 + 1, i *
25     10 + 10))
25
25 # Dodajemo ogranicenja da se u svakoj koloni nalaze razlicite vrednosti
27 for i in range(1, 10):
27     problem.addConstraint(constraint.AllDifferentConstraint(), range(10 + i, 100 + i,
29     10))
29
29 # Dodajemo ogranicenja da svaki podkvadrat od 3x3 promenljive ima razlicite vrednosti
31 for i in [1,4,7]:
31     for j in [1,4,7]:
33         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
33         +2,10*(i+2)+j,10*(i+2)+j+1,10*(i+2)+j+2]
35         problem.addConstraint(constraint.AllDifferentConstraint(),pozicije)
35
37 ime_datoteke = raw_input("Unesite ime datoteke sa tablom za sudoku: ")
37 f = open(ime_datoteke, "r")
39 tabla = json.load(f)
39 f.close()
41
41 # Dodajemo ogranicenja za svaki broj koji je zadat na tabli
43 for i in range(9):
43     for j in range(9):
45         if tabla[i][j] != 0:
45             def o(vrednost_promenljive, vrednost_na_tabli = tabla[i][j]):
47                 if vrednost_promenljive == vrednost_na_tabli:
47                     return True
49
49                 problem.addConstraint(o, [((i+1)*10 + (j+1))])
51
51 resenja = problem.getSolutions()
53
53 for r in resenja:
55     print "======"
55     for i in range(1,10):
57         print "|",
57         for j in range(1,10):
59             if j%3 == 0:
59                 print str(r[i*10+j])+" |",
61             else:
61                 print str(r[i*10+j]),
63         print ""
63         if i%3 == 0 and i!=9:
65             print "-----"
65     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 Korišćenje kompilatora i interpretera.

```
2  -- Ovako pisemo jednolinijske komentare
3  {-
4     Ovako pisemo
5     viselinijске
6     komentare
7  -}
8
9  {-
10     Interpreter pokrecemo iz terminala komandom:
11
12     ghci
13
14     i otvorice nam se interaktivni interpreter:
15
16     GHCi, version 8.0.1: http://www.haskell.org/ghc/  :? for help
17     Prelude>
18
19     Sa interpreterom mozemo direktno komunicirati, npr
20     Prelude> print "Zdravo! :)"
21     "Zdravo! :)"
22
23     Interpretatoru dodajemo mogucnost izvorsavanja funkcija iz izvorne
24     ime_programa.hs komandom:
25
26     :load ime_programa.hs
27
28     Nakon toga mozemo pokrenuti sve funkcije koje su u toj datoteci
29     definisane, npr.
30
31     Prelude> main
32     "Zdravo! :)"
33
34     Iz interpretera se izlazi komandom :quit
35     Prelude> :quit
36
37     Haskell programe mozemo kompajlirati komandom:
38
39     ghc ime_programa.hs
```

3 Funkcionalno programiranje

```
40     koja ce napraviti izvrsni program koji pokrecemo sa
42     ./ime_programa
44 -}
46 main = print "Zdravo! :)"
```

Zadatak 3.2 Tipovi podataka. Tipski razredi. Funkcije.

```
1 {-
2   Osnovni tipovi:
3   Bool
4   Char
5   String
6   Int
7   Integer
8   Float
9   Double
11
12  Tipski razredi (definisu funkcije koje neki tip mora da implementira):
13  Eq - tipovi sa jednakoscu (==,/=)
14  Ord - tipovi sa uredjenjem (<,<=,>,>=,...)
15  Num - numericki tipovi (+,-,*,...)
16  Integral - celobrojni tipovi (div, mod,...)
17  Fractional - razlomacki tipovi (/ , recip,...)
18
19  Funkcije preslikavaju vrednosti jednog tipa (ili tipskog razreda) u vrednost drugog
20  tipa (ili tipskog razreda):
21
22  duplo :: Int -> Int
23
24  Informacije o bilo kom objektu mozete dobiti naredbom:
25  :info ime_objekta
26 -}
27
28 duplo :: Int -> Int
29 duplo x = x+x
30
31 {-
32   Aritmeticke operacije nad tipovima Int, Integer, Float:
33   +, -, *, /,
34   ^^ - stepen (tip Int)
35   ** - stepen (tip Float)
36
37   Funkcije:
38   mod, div, log, sqrt, sin, cos, tan
39 -}
40
41 ostatak3 x = mod x 3
42 -- ekvivalentno sa x `mod` 3
43
44 -- Blokovi se u Haskelu odvajaju tabulatorom.
45 -- Uslovni izraz (mora imati else granu)
46
47 jedan n = if n == 1
48           then True
49           else False
50
51 -- Ogradjene jednacine (guarded equations) - mozemo ih koristiti umesto uslovnih
52   izraza, bitan je redosled navodjenja
53
54 sumaPrvih n
55   | n == 0 = 0
56   | otherwise = n + sumaPrvih(n-1)
```

Zadatak 3.3 Liste. N-torke.

```
1 {-
2   Lista je niz vrednosti istog tipa:
```



```

4  [element1,element2,...]
6  Operacije:
8  x : lista - dodaje element x na pocetak liste
   lista1 ++ lista1 - nadovezuje dve liste
10 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)
12 -}
14 lista = [1..5]
   obrnutaLista = [5,4..1]
16 {-
18 Korisne funkcije:
   head lista - vraca prvi element liste
   tail lista - vraca rep liste
20 length lista - vraca broj elemenata liste
   take n lista - vraca listu prvih n elemenata
22 drop n lista - vraca listu bez prvih n elemenata
   null lista - vraca True ukoliko je lista prazna, False inace
24 -}
26 -- Niske su liste karaktera
28
30 niska1 = ['A','l','a','d','d','i','n']
   niska2 = "Aladdin"
32 {-
   N-torke su nizovi vrednosti razlicitog tipa:
34 ('a',1,[1.2, 2.3],"torka")
36
   Funkcije definisane za parove (n=2):
   fst () - vraca prvi element
   snd () - vraca drugi element
38 -}

```

3.1.2 Zadaci za samostalni rad sa rešenjima

Zadatak 3.4 Napisati sledeće funkcije:

- proizvodPrvih n - računa proizvod prvih n pozitivnih brojeva (rekurzivno, bez korišćenja formule)
- prost n - vraća True ako je broj prost, False inače
- nzd a b - računa najmanji zajednički delilac brojeva a i b (koristiti Euklidov algoritam)
- tipJednacine a b c - vraća tip kvadratne jednačine $a * x^2 + b * x + c = 0$ (degenerisana”, ”jedno resenje”, dva resenja”, ”bez resenja”)
- izDekadne x osn - prebacuje broj x iz dekadne u osnovu osn (pretpostaviti da je osn>1 i osn<10)
- uDekadne x osn - prebacuje broj x iz osnove osn u dekadnu osnovu (pretpostaviti da je osn>1 i osn<10)
- ceoDeo x - računa ceo deo korena broja x (bez korišćenja ugrađenih funkcija za koren i/ili stepen)

[Rešenje 3.4]

Zadatak 3.5 Napisati sledeće funkcije koristeći liste:

- harm n - vraća listu prvih n elemenata harmonijskog reda
- delioci n - vraća listu svih delilaca pozitivnog broja n

- nadovezi lista1 lista2 n - nadovezuje na lista1 n puta lista2

[Rešenje 3.5]

3.1.3 Zadaci za vežbu

Zadatak 3.6 Napisati sledeće funkcije:

- sumaKvadrata n - računa sumu kvadrata prvih n brojeva (rekurzivno, bez korišćenja formule)
- brojDelilaca n - vraća broj delilaca broja n
- fib n - računa n-ti element Fibonačijevog reda
- osnova x osn1 osn2 - prebacuje broj x iz osnove osn1 u osnovu osn2 (pretpostaviti da su osn1 i osn2 brojevi veći od 1 i manji od 10)

Zadatak 3.7 Napisati sledeće funkcije koristeći liste:

- parni n - vraća listu prvih n parnih brojeva
- fibLista n - vraća listu prvih n elemenata Fibonačijevog niza
- jednocifreniDeliooci n - vraća listu svih jednocifrenih delilaca broja n

3.2 Liste

3.2.1 Uvodni primeri

Zadatak 3.8

```
1 {-
2   Uparivanje sablona (pattern matching)
3
4   not :: Bool -> Bool
5   not False = True
6   not True = False
7
8   Ako pozovemo not True uparice se prvi sablon,
9   a ako pozovemo not False uparice se drugi sablon
10
11  Dzoker (wildcard)
12
13  Prvi nacin:
14  (&&) :: Bool -> Bool -> Bool
15  True && True = True
16  _ && _ = False
17
18  Moze i:
19  n && m = False
20  Bitno je razlicito imenovanje izraza n i m
21
22  Ukoliko bismo zamenili redosled sablona, javila bi se greska "Pattern match
23  overlapped"
24
25  Drugi nacin, efikasniji:
26  (&&) :: Bool -> Bool -> Bool
27  True && x = x
28  False && _ = False
29
30  _ predstavlja dzoker, tj. sablon koji odgovara bilo kojoj vrednosti (obratiti
31  paznju na redosled definisanja sablona!)
32
33  Sabloni lista:
34
35  length [_] = 1 - [_] predstavlja listu sa jednim elementom
```

```

length x = 1 + length (tail x) - x predstavlja listu
35 -}
37
length1 [] = 0
39 length1 x = 1 + length1 (tail x)

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

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

45 -- preslikava iz liste elemenata u jedan element
47 glava :: [a] -> a
glava (x:_) = x
49

-- preslikava iz liste u listu
51 rep :: [a] -> [a]
rep (_:xs) = xs
53

-- restrikcija na liste elemenata koji pripadaju numerickim tipovima
55 glavNum :: (Num a) => [a] -> a
glavNum (x:_) = x
57

-- Generatori liste (list comprehension)
59

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

65 deljiviPet = [x | x<-[1..50], mod x 5 == 0] -- elementi liste pripadaju skupu od 1 do
50 i deljivi su sa 5

67 kvadrati = [x^2 | x<-[1..10]]

69 -- Mozemo imati vise generatora

71 torke1 = [(x,y) | x<-[1..5], y<-[6..9]]

73 -- obratiti paznju kako utice zamena mesta generatora
torke2 = [(x,y) | y<-[6..9], x<-[1..5]]
75

-- sa negativnim korakom
77 torke3 = [(x,y) | y<-[9,8..6], x<-[1..5]]

79 -- Generatori mogu zavisiti jedni od drugih

81 -- kasniji generatori mogu zavisiti samo od promenljivih koje su uvedene ranijim
generatorima citajuci sa leva na desno
torke4 = [(x,y) | x<-[1..10], y<-[x..10]]
83

{-
85 Sekcija where

87 where ime = vrednost
- imena koja definisemo u where sekciji su vidljiva samo u okviru funkcije
89 u kojoj su definisana
-}

91
bmiIzracunaj tezina visina
93 | bmi <= 18.5 = "Mrsavko"
| bmi <= 25.0 = "Sportista"
95 | bmi > 25.0 = "Bucko"
where bmi = tezina / visina ^ 2

```

3.2.2 Zadaci za samostalni rad sa rešenjima

Zadatak 3.9 Napisati sledeće funkcije:

- bezbedanRep lista - ukoliko je lista prazna vraća praznu listu, inače vraća rep liste, kori-

steći: a) uslovne izraze b) ograđene jednačine c) uparivanje šablona

- `savršeni n` - vraća listu savršenih brojeva manjih od n (broj je savršen ako je jednak sumi svojih faktora (tj. delilaca), ne uključujući taj broj)
- `zbirPar n` - vraća listu parova (a,b) takvih da su a i b prirodni brojevi i da je zbir a i b jednak n
- `polovina lista` - deli listu na dve polovine
- `poslednji lista` - vraća poslednji element liste
- `spoji lista` - spaja listu listi u jednu listu
- `sufiksi lista` - vraća listu svih sufixa liste
- `obrni lista` - obrće listu
- `izbaci k lista` - izbacuje k -ti element iz liste
- `duplira lista` - duplira svaki element iz liste
- `ubaci k x lista` - ubacuje u listu na poziciju k element x
- `izbaciSvaki n lista` - izbacuje svaki n -ti element iz liste (broji se od 1)

[Rešenje 3.9]

3.2.3 Zadaci za vežbu

Zadatak 3.10 Napisati sledeće funkcije:

- `prosti n` - vraća listu prvih n prostih brojeva
- `pitagorineTrojke n` - vraća listu Pitagorinih trojki (a,b,c) takvih da su $a,b,c \leq n$
- `proizvodPar n` - vraća listu parova (a,b) takvih da su a i b prirodni brojevi i da je proizvod a i b manji od n
- `podeli n lista` - deli listu na dve liste, prva lista sadrži prvih n elemenata, a druga lista sadrži ostatak
- `pretposlednji lista` - vraća pretposlednji element liste
- `izbrisiPrvi x lista` - briše prvo pojavljivanje broja x u listi
- `palindrom lista` - ispituje da li je lista palindrom
- `pozNeg lista` - vraća par (a, b) takav da a predstavlja broj `True` elemenata u listi, a b predstavlja broj `False` elemenata u listi
- `izmedju a b lista` - vraća listu elemenata koji su veći od a , a manji od b

3.3 Rešenja

Rešenje 3.4

```

1  -- Pretpostavljamo da funkciju koristimo samo za n>=1
   proizvodPrvih n
3     | n == 1 = 1
     | otherwise = n * proizvodPrvih (n-1)
5
6  -- Ispitujemo da li je broj prost tako sto pozovemo pomocnu funkciju
7  -- koja ispituje da li postoji neki broj pocev od 2 do n koji deli broj n
   prost n = prost1 n 2
9
10 -- Proveravamo da li postoji broj koji deli n (pocev od broja 2 - poziv iz prethodne
    funkcije)
11 -- 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
13 -- 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)
15 prost1 n k
    | n == k = True
17   | n `mod` k == 0 = False
    | otherwise = prost1 n (k+1)
19
20 nzd a b
21   | b == 0 = a
    | otherwise = nzd b (a `mod` b)
23
24 tipJednacine a b c
25   | a == 0 = "Degenerisana"
    | (b*b - 4*a*c) == 0 = "Jedno resenje"
27   | (b*b - 4*a*c) > 0 = "Dva resenja"
    | otherwise = "Nema resenja"
29
30 uDekadnu x osn = if x==0 then 0 else uDekadnu (x `div` 10) osn * osn + (mod x 10)
31
32 izDekadne x osn = if x==0 then 0 else izDekadne (x `div` osn) osn * 10 + (mod x osn)
33
34 -- Trazimo ceo deo korena broja x
35 -- 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
37 ceoDeo x = ceoDeo1 x 1
39
40 ceoDeo1 x i
    | (i*i) > x = (i-1)
41   | otherwise = ceoDeo1 x (i+1)

```

Rešenje 3.5

```

1  -- Pravimo listu prvih n elemenata harmonijskog reda (n>0)
    -- ukoliko je n = 1, vracamo listu koja sadrzi prvi element harmonijskog reda
3  -- inace pozovemo funkciju rekurzivno za n-1
    -- i na rezultat nadovezemo reciprocnu vrednost broja n
5  harm n
    | n == 1 = [1.0]
7   | otherwise = harm (n-1) ++ [recip n]
9
10 -- Pravimo listu delioca broja n
    -- ukoliko je n = 1, vracamo listu koja sadrzi samo 1
11 -- inace pozivamo pomocnu funkciju sa jos jednim parametrom
    -- koji nam govori do kog potencijalnog delioca smo stigli
13 delioci n
    | n == 1 = [1]
15   | otherwise = delioci1 n 2
17
18 -- Pravimo listu delioca broja n pocev od broja k
    -- ukoliko smo stigli do broja n (k==n) vracamo praznu listu
19 -- inace proveravamo da li je k delioc broja n
    -- ako jeste, stavljamo ga u listu i pozivamo funkciju rekurzivno
21 -- za k+1 (sledeci potencijalni delilac)
    -- inace samo pozivamo funkciju rekurzivno za k+1
23 delioci1 n k
    | k == n = []
25   | n `mod` k == 0 = [k] ++ (delioci1 n (k+1))

```

```
    | otherwise = delioci1 n (k+1)
27
-- Nadovezujemo listu2 na listu1 n puta
29 -- ukoliko je n == 1 na listu1 nadovezemo listu2 operatorom ++
-- inace pozovemo funkciju rekurzivno za n-1 i na rezultat nadovezemo listu2
31 nadovezi lista1 lista2 n
    | n == 1 = lista1 ++ lista2
33 | otherwise = (nadovezi lista1 lista2 (n-1)) ++ lista2
```

Rešenje 3.9

```
bezbedanRep1 lista = if lista == [] then [] else tail lista
2
bezbedanRep2 lista
4 | lista == [] = []
  | otherwise = tail lista
6
bezbedanRep3 [] = []
8 bezbedanRep3 (x:xs) = xs
-- bezbedanRep3 lista = tail lista
10
savršeni n = [x | x<-[1..n], sum(faktori x) == x]
12 faktori x = [i | i<-[1..x-1], x `mod` i ==0]
14
-- Prvi nacin
zbirPar1 n = [(a,b) | a<-[1..n], b<-[1..n], a+b==n]
16 -- Drugi nacin, efikasniji
zbirPar2 n = [(a,b) | a<-[1..n], b<-[n-a], b/=0]
18
polovina [] = ([],[])
20 polovina lista = (take n lista, drop n lista)
                    where n = length lista `div` 2
22
poslednji lista = lista !! poz
24                    where poz = length lista - 1
26
spoji [] = [] -- nije neophodno
spoji lista = [x | podlista <- lista, x <- podlista]
28
-- Lista je sama svoj sufiks, a ostali sufiksi su sufiksi njenog repa
30 sufiksi [] = [[]]
sufiksi (x:xs) = (x:xs) : sufiksi xs
32
obrni [] = []
34 obrni (x:xs) = obrni xs ++ [x]
36
izbaci _ [] = []
izbaci 0 (x:xs) = xs
38 izbaci k (x:xs) = x : (izbaci (k-1) xs)
40
duplira [] = []
duplira (x:xs) = [x,x] ++ duplira xs
42
ubaci 0 n lista = n : lista
44 ubaci k n [] = [n]
ubaci k n (x:xs) = x : (ubaci (k-1) n xs)
46
izbaciSvaki n [] = []
48 izbaciSvaki n lista = take (n-1) lista ++ izbaciSvaki n (drop n lista)
```