

## Vaje za ponavljanje - 2. del

Tam, kjer se ti zdi primerno, napiši svoje *funkcije*, tudi če naloga tega ne zahteva eksplicitno :-)

### 1. naloga: Podčrtavanje

Napiši program, ki prebere besedilo z zaznamki za podčrtavanje in ga nato ustrezno oblikovanega izpiše. Tisti del besedila, ki naj bo izpisan podčrtano, je na začetku in koncu označen s podčrtajem “\_”. Črto pod besedilom oblikuj v naslednji vrstici s črticami “-”. V vhodnem besedilu bo vedno sodo mnogo podčrtajev. (Vsak zaznamek, ki se začne, se tudi konča; znak “\_” ne nastopa v drugi vlogi, kot začetek ali konec zaznamka.)

Primeri #1:

```
Vnesi besedilo: Te naloge so _zelo lahke_. Kakor za koga.  
Te naloge so zelo lahke. Kakor za koga.
```

```
-----
```

Primer #2:

```
Vnesi besedilo: _Nekatere_ je mogoce celo _resiti_.  
Nekatere je mogoce celo resiti.
```

```
-----
```

### 2. naloga: Asterix (in drugi Galci)

Če ti je kdaj pod roke prišel kakšen strip z Asterixom<sup>1</sup> in Obelixom, veš, da se imena vseh Galcev končajo na “ix”: Asterix, Obelix, Panoramix, Fisteltenorkix in tako naprej.

Napiši program, ki prebere ime, nato pa izpiše “Galec!”, če gre za galško ime (torej tako, ki se konča na “ix”), v nasprotnem primeru pa izpiše “Rimljan!”. (Vsa vnešena imena bodo dolga najmanj 3 znake.)

Primer #1:

```
Vnesi ime: Ataaufbix  
Galec!
```

Primer #2:

```
Vnesi ime: Zlatko  
Rimljan!
```

### 3. naloga: Registrske tablice

Napiši program, ki preveri veljavnost avtomobilske registrske tablice. Veljavne so tiste tablice, ki ustrezajo naslednjim pogojem:

---

<sup>1</sup>Glej: <http://en.wikipedia.org/wiki/Asterix>

- prva dva znaka tablice sta oznaka mesta, torej: LJ, MB, CE, GO, NM, KP, KK, KR, PO, SG ali MS
  - tretji znak je presledek (namesto grba)
  - presledku sledi najmanj 3 in največ 6 znakov;
  - ti znaki so lahko velike tiskane angleške črke in številke, tretji od teh znakov pa je lahko tudi črtica
- Program naj prebere tablico in izpiše “Veljavna!” ali “Neveljavna!”.

Primer:

Vnesi tablico: LJ 35-AF8

Veljavna!

#### 4. naloga: **KiKoDa**

Putka Gertruda je znana borka za zborna govorica v kokošji populaciji. Veliko napora vlaga zlasti v osveščanje perjadi o kulturi uporabe vejice, tako v pisni kot tudi v govorni obliki. Zdi se ji nekako nedopustno, da bi lepo vzgojene putke pozabile na vejice, še zlasti pred besedami: “ki”, “ko”, “da”, saj tvorijo glavnino kokošnjega besedišča. Nujno potrebuje pregledovalnik besedila, prirejen za kokošjo populacijo, ki preveri in po potrebi ustrezno popravi besedilo.

Napiši program, ki prebere eno vrstico besedila, dolgo največ 200 znakov in preveri, če pred besedami: “ki”, “ko”, “da” stoji vejica. Če vejica manjka, jo mora program dodati. Program naj izpiše popravljeno besedilo. (V besedilu se ne bodo nikoli pojavili podvojeni presledki. Prvi in zadnji znak v besedilu ne bosta presledka. Besedilo se nikoli ne bo začelo s “ki”, “ko” ali “da”.)

Primeri #1:

Vnesi besedilo: Poklical te bom ko bom doma.

Poklical te bom, ko bom doma.

Primer #2:

Vnesi besedilo: Pohiteti moramo, da bomo do kosila doma.

Pohiteti moramo, da bomo do kosila doma.

#### 5. naloga: **Smith & Wesson**

Znak “&” se pogosto uporablja namesto besede “in”. (Pravzaprav je nastal tako, da so ljudje vedno bolj skupaj in vedno bolj umetelno pisali besedo “et”, to je “in” po latinsko. Zato “&” včasih še vedno izgovorimo “et”.)

Vendar pa tega znaka ne uporabljamo kar vsepovprek – tale stavek, na primer, je videti kaj čudno: *Rdeča kapica je šla k babici & ji nesla potico & grozdje*. Razlog je, da “&” uporabimo predvsem v sestavljenih imenih. Na primer: *Lovec je s svojim Smith & Wesson revolverjem ustrelil volka, objel Rdečo kapico in šel domov gledat Tom & Jerrya*.

Če malo poenostavimo, lahko rečemo, da se da besedo “in” zamenjati z znakom “&”, če se ta “in” nahaja med dvema besedama z veliko začetnico. Napiši program, ki bo prebral stavek, zamenjal “in” z “&” povsod, kjer pravkar opisano pravilo to dovoljuje, in izpisal spremenjeni stavek. (V besedilu se ne bodo nikoli pojavili podvojeni presledki. Prvi in zadnji znak v besedilu ne bosta nikoli presledka.)

Primer:

Vnesi niz: Zajca Rado in Vlado imata rada korenje in kolerabo.  
Zajca Rado & Vlado imata rada korenje in kolerabo.

## 6. naloga: **Viteštvo na Slovenskem**

Knez Trebikor se zaveda, da naše kraje ogrožajo Turki. (Za tiste, ki vam še ni jasno: piše se leto Gospodovo 1474.) Slovenci imamo dovolj vitezov za obrambo pred turško nadlogo, vendar so med seboj sprti. Trebikor bi rad organiziral sestanek, kjer bi pomiril sprte strani. Zbrali se bodo za okroglo leseno mizo. Vitezi se delijo na tri skupine: bele, rdeče in modre. Beli so prijatelji z vsemi drugimi in tudi Trebikor sodi mednje, sprti pa so rdeči in modri.

Napiši program, ki bo prebral niz, ki predstavlja razporeditev vitezov okrog okrogle mize. Beli so označeni z “B”, modri z “M” in rdeči z “R”. (Velike tiskane črke.) Niz se vedno začne pri Trebikorju, torej je prvi znak vedno “B”. Ne pozabi, da je miza okrogla, torej tudi prvi in zadnji sedita skupaj.

Trebikor bi rad določene podatke, ki bi mu bili v pomoč pri vodenju sestanka. Program naj glede na ta niz prešteje bele, modre in rdeče plemiče, ter dolžino najdaljše prijateljske skupine. Prijateljska skupina je skupina vitezov, ki sedijo za mizo zaporedno drug poleg drugega, tako da med njimi ni nobeden z nobenim sprt. BMMBBMMBB je prijateljska skupina, ker so notri samo beli in modri, ki so med seboj prijatelji; BMMBBRBB pa ne, ker je notri tudi rdeč, ki je sprt z modrimi.

Primer:

Kako sedijo vitezi? BMBBRMMBRRBMMMRBRM

St. belih vitezov: 6

St. modrih vitezov: 7

St. rdecih vitezov: 5

Najdaljsa prijateljska skupina: 5

## 7. naloga: **Bonbonjera**

Ko je nekega dne Simon prišel iz šole, je na mizi zagledal bonboniero. Ker je izjemno sladkosned, se ni prav nič obotavljal – toda kaj kmalu je opazil,

da je sestra Urška nekatere bonbone že pojedla. Simon pa (iz nam neznanega razloga) nikoli ne je bonbonov, ki so v isti vrstici ali istem stolpcu v bonbonieri, kakor tisti, ki jih je pojedla Urška. Pomagajmo mu izračunati koliko bonbonov bo lahko pojedel.

Bonboniera je pravokotne oblike. Ima  $V$  vrstic in  $S$  stolpcev. Ko je bila polna, je bilo v njej torej  $V \cdot S$  bonbonov. Vsak bonbon določata dve koordinati in sicer številka vrstice in številka stolpca v katerem se ta nahaja.

Tvoj program naj najprej prebere velikost bonboniere (ta bo imela vedno manj kot 1000 vrstic in manj kot 1000 stolpcev), nato pa naj poizve, koliko bonbonov je pojedla Urška in prebere še koordinate pojedenih bonbonov. Program naj izpiše eno samo število – maksimalno število bonbonov, ki jih lahko poje Simon.

Primer:

```
Kako velika je bonboniera? 10 10
Koliko bonbonov je pojedla Urska? 5
Vnesi koordinate 1. bonbona: 2 2
Vnesi koordinate 2. bonbona: 3 6
Vnesi koordinate 3. bonbona: 7 3
Vnesi koordinate 4. bonbona: 8 5
Vnesi koordinate 5. bonbona: 8 9
Simon lahko poje 30 bonbonov.
```

## 8. naloga: **Fužine**

Na Fužinama je velik blok. Kot po navadi se v takih blokih najdejo nepridipravi, ki poslušajo glasbo preveč naglas. To njihovim sosedom ni po godu. Če sosed stanuje zraven enega nepridiprava je le *slabe volje*, če pa stanuje zraven dveh ali več je pa *jezen*. Nepridipravi niso niti jezni niti slabe volje, tudi če imajo za sosede druge nepridiprave.

Blok si predstavljajmo tako, kot bi ga videli, če bi se postavili na “dvorišče” pred njega (torej bo za nas 2-dimenzionalen). Videli bi več nadstropij, v vsakem po eno vrstico stanovanj. V vsakem nadstropju je enako število stanovanj, ki so vsa enako velika. Ostala stanovanja, ki niso vidna z “dvorišča” bloka nas ne zanimajo. Za sosede enega stanovalca se smatra vse sosede okoli njega. Običajna stanovanja imajo po 8 sosedov. Izjema so stanovanja, ki so v pritličju, potem tista v najvišjem nadstropju in pa prva in zadnja stanovanja v nadstropju. Taka imajo 5 ali pa 3 sosede.

Napiši program, ki te vpraša kako velik je blok, koliko je nepridipravov in kje ti stanujejo. Prva vnesena številka pomeni nadstropje, druga pa številko stanovanja. Stanovanja so oštevilčena od leve proti desni s števili od 1 naprej. Noben blok ne bo imel več kot 50 nadstropij in ne bo širši od 50 stanovanj. Program naj izpiše število stanovalcev, ki so jezni in število stanovalcev, ki so samo slabe volje.

Primer<sup>2</sup>:

Koliko nadstropij ima blok? 12  
Koliko stanovanj je v nadstropju? 14  
Koliko nepridipravov zivi v bloku? 3  
Kje zivi 1. nepridiprav? 3 5  
Kje zivi 2. nepridiprav? 11 3  
Kje zivi 3. nepridiprav? 1 4

St. jeznih stanovalcev: 2  
St. stanovalcev slabe volje: 17

### 9. naloga: **Encyclopaedia Podgannica**

Slabo znano dejstvo je, da so najinteligentnejša bitja na svetu podgane. Ravnokar se pripravljajo, da bi izdale knjigo, v kateri bi bilo zajeto vse njihovo dosedanje znanje. Encyclopaedia Podgannica, kot se bo knjiga imenovala, bo imela ogromno strani (vendar manj kot 100000), zato podgane zanima, koliko katerih števk bodo potrebovale, da strani knjige oštevilčijo. Napiši program, ki vpraša, koliko strani bo imela nova knjiga, nato pa izpiše, koliko posameznih števk bodo podgane rabile za oštevilčevanje strani knjige.

Primer:

Koliko strani bo imela enciklopedija? 14

1-krat 0  
7-krat 1  
2-krat 2  
2-krat 3  
2-krat 4  
1-krat 5  
1-krat 6  
1-krat 7  
1-krat 8  
1-krat 9

### 10. naloga: **Pregljeva**

Na jugovzhodu Ljubljane leži Pregljeva ulica, ob kateri stoji zelo veliko hiš, saj od glavnega dela ulice vodijo do posameznih hiš manjši dovozi brez imen. Hišne številke so hišam podeljene takole: na levi strani imajo hiše lihe številke, in sicer ob prvem dovozu od 1 do 20, ob drugem dovozu od 21 do 40 itd. Podobno je vse skupaj na desni strani, le da so tam uporabljene sode številke.

---

<sup>2</sup>Prvi nepridiprav živi v 3. nadstropju v 5. stanovanju, drugi nepridiprav v 11. nadstropju v 3. stanovanju, itd.

Napiši program, ki prebere hišno številko, nato pa izpiše, kje na Pregljevi ta hiša leži. Za obliko izpisa glej spodnje primere. Seveda Pregljeva ulica ni neskončno dolga – na njej je 400 hiš. Če program dobi v obdelavo številko, ki je na Pregljevi ni, naj izpiše sporočilo o napaki.

Primer #1:

```
Vnesi hisno številko: 41
Hisa s hisno st. 41 je 1. v 3. vrsti na levi strani.
```

Primer #2:

```
Vnesi hisno številko: 24
Hisa s hisno st. 24 je 2. v 2. vrsti na desni strani.
```

Primer #3:

```
Vnesi hisno številko: 524
Hise s hisno st. 524 ni v Pregljevi ulici.
```

### 11. naloga: **Pascalov trikotnik**

Napiši program, ki te bo vprašal po številu vrstic, nato pa bo izpisal Pascalov trikotnik<sup>3</sup> z željenim številom vrstic.

Primer:

```
Vnesi stevilo vrstic: 9
```

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
```

### 12. naloga: **Deljivost velikih števil**

Veliko naravno število lahko predstavimo z nizom njegovih števk. Napiši program, ki bo prebral niz, v katerem je zapisano naravno število, ter preveril, ali je to število deljivo s 3, 5, 9 ali 11. Število je:

- deljivo s 3, če je vsota njegovih števk deljiva s 3;
- deljivo s 5, če se konča z 0 ali 5;
- deljivo z 9, če je vsota njegovih števk deljiva z 9;

---

<sup>3</sup>Glej: [http://en.wikipedia.org/wiki/Pascal's\\_triangle](http://en.wikipedia.org/wiki/Pascal's_triangle)

- deljivo z 11, če je alternirajoča<sup>4</sup> vsota njegovih števk deljiva z 11.

Primer:

Vnesi celo stevilo: 895593020580193742536011937017

Stevilo je deljivo s 3.

Stevilo ni deljivo s 5.

Stevilo ni deljivo z 9.

Stevilo je deljivo z 11.

### 13. naloga: **Permutiraj opico**

Napiši program, ki prebere kratek niz (največ 9 znakov) in celo število enake dolžine, ki predstavlja permutacijo. Program naj sestavi in izpiše permutiran niz glede na dano število.

Primer:

Vnesi niz: opica

Vnesi permutacijo dolžine 5: 15234

Permutiran niz je: oapic

### 14. naloga: **Rimski imperij vrača udarec**

Napiši program, ki te bo vprašal po nekem številu, nato pa bo to število izpisal v rimskem<sup>5</sup> zapisu. Program naj deluje za števila med 1 in 3999.

Primer:

Vnesi stevilo: 1986

Rimski zapis: MCMLXXXVI

### 15. naloga: **Vrnitev odpisanih števil**

Napiši program, ki bo povprašal po nekem rimskem številu, nato pa bo to število izpisal v arabskem<sup>6</sup> zapisu. Si lahko kako pomagaš z rešitvijo prejšnje naloge?

Primer:

Vnesi rimsko stevilo: MCMLXXXVI

Arabski zapis: 1986

### 16. naloga: **Kralji**

Ker se imena kraljev v posameznih kraljevinah pogosto ponavljajo, se njihovim imenom doda še številka, da se natančno ve, za katerega kralja gre. V angleški zgodovini je bilo kar nekaj kraljev z imenom Henry (Henry I, Henry II, ..., Henry VIII).

---

<sup>4</sup>Alternirajočo vsoto dobimo tako, da izmenično prištevamo in odštevamo števke.

<sup>5</sup>Glej: [http://en.wikipedia.org/wiki/Roman\\_numerals](http://en.wikipedia.org/wiki/Roman_numerals)

<sup>6</sup>Glej: [http://en.wikipedia.org/wiki/Arabic\\_numerals](http://en.wikipedia.org/wiki/Arabic_numerals)

Iz *datoteke* preberi podatke o vladarjih. V prvi vrstici je število vladarjev, tej pa sledijo vsa njihova imena. Izpiši jih v istem vrstnem redu. Imenom, ki se ponovijo večkrat, pa dodaj še zaporedno številko. Predpostaviš lahko naslednje:

- vsa imena bodo zapisana z mali črkami, le začetnica bo velika;
- vse imena bodo enobesedna;
- imen ne bo več kot 20.

Primer vhodne datoteke:

```
10
Edward
Mary
Henry
Richard
Henry
Henry
Charles
William
Charles
Edward
```

Pripadajoča izhodna datoteka:

```
Edward 1
Mary
Henry 1
Richard
Henry 2
Henry 3
Charles 1
William
Charles 2
Edward 2
```

### 17. naloga: **Zaokroževanje**

Ljudje radi zaokrožujemo števila na večkratnike števila 10. Namesto 48 km/h pogostokrat raje rečemo 50 km/h in namesto 32 kg včasih rečemo kar 30 kg. Tudi putke zelo rade zaokrožujejo števila, vendar nikakor ne na večkratnike števila 10. Veliko raje imajo druga števila. Še bolj hecno pa je to, da ima vsaka putka svoje število, na katerega najraje zaokrožuje. Tvoja naloga je, da putkam pomagaš zaokroževati njihova števila. Vsaka ti bo povedala njeno najljubše število za zaokroževanje, ter še nekaj števil, ki jih boš moral zaokrožiti.

Tvoj program naj podatke prebere iz *datoteke*. V prvi vrstici datoteke bo število  $K$  ( $2 \leq K \leq 60$ ).  $K$  bo venomer različen od 10. Od druge vrstice

dalje bodo števila med 1 in 10000, ki jih je potrebno zaokrožiti. Program naj v neko drugo *datoteko* izpiše števila (vsako v svojo vrstico) zaokrožena na večkratnike števila  $K$ .

Primer vhodne datoteke:

```
6
6
27
19
4
9
24
17
24
23
13
```

Pripadajoča izhodna datoteka<sup>7</sup>:

```
6
30
18
6
12
24
18
24
24
12
```

#### 18. naloga: **Palindromski kvadrati**

Napiši program, ki bo izpisal tiste kvadrate števil med 1 in  $n$ , ki so palindromi<sup>8</sup>. Število  $n$  je zgornja meja, ki jo preberemo preko tipkovnice.

Primer:

Vnesi zgornjo mejo: 20

Palindromski kvadrati: 1 4 9 121

Kvadrati števil med 1 in 20 so: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361 in 400. Med njimi so palindromi zgolj: 1, 4, 9 in 121.

---

<sup>7</sup>Kadar zaokrožuješ število, ki je ravno na sredini med dvema večkratnikoma, zaokroži navzgor. V zgornjem primeru smo tako pri zaokroževanju na večkratnike števila 6 število 27 zaokrožili na 30 in ne na 24.

<sup>8</sup>Palindrom je beseda, ki se bere enako z leve in z desne.

### 19. naloga: **Srečna števila**

Napiši program, ki izračuna in izpiše vsa srečna števila, ki so manjša od 1000. Srečna števila so tista, ki preživijo naslednji postopek:

V vrsto postavimo vsa števila med 1 in  $n$ . 1 izpustimo in začnemo z 2. Iz vrste izločimo vsako drugo število. Naslednje število, ki je večje od 2 je 3. Zda j izločimo vsako tretje število. Naslednje število v vrsti, ki je večje od 3 je 7. Izločimo vsako sedmo število. Postopek ponavljamo, dokler lahko izločimo še kakšno število. Števila vedno štejemo od začetka vrste, tj. od 1 dalje. Primer (za  $n = 24$ ):

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24  
1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23  
1, 3, 7, 9, 13, 15, 19, 21  
1, 3, 7, 9, 13, 15, 21

### 20. naloga: **Naša četica koraka**

Napiši funkcijo `dolzineCet`, ki bo kot argument dobila neko tabelo ter njeno dožino in izračunala dolžine čet<sup>9</sup> v tej tabeli. Funkcija naj vrne novo tabelo, katere prvi element je število čet, vsi nadaljnji elementi pa so njih dolžine. Primer:

V tabeli [3, 8, 9, 2, 1, 8, 7, 9, 9, 10, 11, 3, 2] imamo naslednje čete:

3, 8, 9  
2  
1, 8  
7, 9, 9, 10, 11  
3  
2

V tem primeru bi funkcija vrnila tako tabelo: [6, 3, 1, 2, 5, 1, 1].

S funkcijo `main` preveri delovanje te funkcije.

### 21. naloga: **Tabelna fuzija**

Sestavi funkcijo `zlijTabeli`, ki dani tabeli zlije v eno. V novo tabelo se izmenično zapisujejo elementi iz prve in druge tabele, začeni s prvo. Če v kaki tabeli zmanjka elementov, se zapišejo še vsi preostali elementi iz daljše tabele. Iz tabel [0, 9, 4] in [1, 3, 5, 6] bi dobili tabelo [0, 1, 9, 3, 4, 5, 6].

S funkcijo `main` preveri delovanje te funkcije.

### 22. naloga: **Ciklični pomik tabele**

Sestavi funkcijo `ciklicniPomik`, ki sprejme tri argumente. Prvi argument je tabela, drugi je njena dolžina tretji pa je celo število  $k$ . Funkcija naj vrne

<sup>9</sup>Četa je vsako maksimalno strnjeno nepadajoče zaporedje elementov tabele.

ново tabelo, ki bo krožni pomik podane tabele in sicer za  $k$  mest v desno. Na primer iz tabele [1, 2, 3, 4, 5, 6, 7, 8] bi pri  $k = 3$  funkcija naredila tabelo [6, 7, 8, 1, 2, 3, 4, 5].

S funkcijo `main` preveri delovanje te funkcije.

### 23. naloga: **Slovar**

V neki *datoteki* imamo zapisane besede, urejene po abecedi. Običajno se nekaj začetnih črk prejšnje besede ponovi v naslednji. Da bi prihranili prostor, so namesto teh črk pred preostali del besede napisali cifro med 2 in 9, ki pomeni, koliko začetnih črk je treba prevzeti od prejšnje besede. Če pred besedo ni cifre, pomeni to isto kot 0. Vsaka beseda je zapisana v svoji vrstici datoteke.

Ta datoteka je nastala v starih časih, ko so bili diski majhni in so ljudje intenzivno “šparali”. Dandanes pa ni več potrebe, da bi bile besede tako zakodirane. Sestavi program, ki bo besede razkodiral in jih zapisal v novo datoteko. Predpostaviš lahko, da so uporabljene samo majhne črke angleške abecede.

Primer vhodne datoteke:

```
abeceda
6en
6nik
2normalen
8no
9ost
```

Pripadajoča izhodna datoteka:

```
abeceda
abeceden
abecednik
abnormalen
abnormalno
abnormalnost
```

### 24. naloga: **Krčenje nizov**

Sestavi funkcijo, ki bo kot argument dobila niz znakov in vrnila nov niz, ki ga dobimo iz danega niza na naslednji način:

Znake po vrsti prepisujemo iz enega niza v drugega, če pa se pojavita zapovrstjo dva enaka znaka, ju nadomestimo z naslednjim znakom v abecedi. (Uporabljamo seveda angleško abecedo.) Tako “aa” nadomestimo z “b”, “bb” nadomestimo s “c”, itd. Izjema je “zz”, ki nima nasednje črke, zato ta par nadomestimo z “a”. Iz “mississippi” bi po tem postopku dobili “mititiqi”.

Izkaže se, da se pri nekaterih besedah po obdelavi spet nastanejo podvojene črke. Taka beseda je npr. “pool”. Iz nje dobimo besedo “ppl”. To pa lahko še enkrat skrčimo, pa dobimo “ql”. Napiši še eno funkcijo, ki bo niz skrčila do konca. Ta naj kliče prejšnjo funkcijo, ki naredi en korak v tem procesu.

S funkcijo `main` preveri delovanje obeh funkcij.

## 25. naloga: **Agilnost**

Napiši funkcijo, ki bo kot argument dobila niz znakov in vrnila nov niz, v katerem bodo znaki prvotnega niza urejeni po velikosti.

Nato sestavi funkcijo, ki bo kot argument dobila niz in preverila, če je ta slučajno naraščajoča beseda. Naraščajoča beseda je tista, v kateri je vsaka naslednja črka “večja” od prejšnje (tj. v abecedi se pojavi kasneje). Primer take besede je “agilnost”.

Sestavi še funkcijo, ki bo sprejela dva argumenta, ki bosta oba niza znakov. Funkcija naj vrne celo število 1, če sta niza anagrama<sup>10</sup> in 0, če nista. (V C-ju logične vrednosti “fejkamo” s celimi števili in sicer 0 pomeni neresnično, 1 pa resnično vrednost.)

S funkcijo `main` preveri delovanje teh funkcij.

*Chuck Norris can create a rock so heavy that  
even he can't lift it. And then he lifts it anyways,  
just to show you who the fuck Chuck Norris is.*

---

<sup>10</sup>Dve besedi sta anagrama, če sta iz istih črk, toda ne nujno v enakem vrstnem redu. Primer: “nadrealist” in “destilarna”.