16. tekmovanje ACM v znanju računalništva za srednješolce

27. marca 2021

NALOGE ZA PRVO SKUPINO

Naloge rešuj samostojno; ne sprašuj drugih ljudi za nasvete ali pomoč pri reševanju (niti v živo niti prek interneta ali kako drugače), ne kopiraj v svoje odgovore tuje izvorne kode in podobno. Tekmovalna komisija si pridržuje pravico, da tekmovalca diskvalificira, če bi se kasneje izkazalo, da nalog ni reševal sam. Internet lahko uporabljaš, če ni v nasprotju s prejšnjimi omejitvami (npr. za branje dokumentacije), vendar za reševanje nalog ni nujno potreben. Tvoje odgovore bomo pregledali in ocenili ročno, zato manjše napake v sintaksi ali pri klicih funkcij standardne knjižnice niso tako pomembne, kot bi bile na tekmovanjih z avtomatskim ocenjevanjem.

Tekmovanje bo potekalo na strežniku `https://rtk.fri.uni-lj.si/`, kjer dobiš naloge in oddajaš svoje odgovore. Uporabniška imena in gesla (bo)ste dobili po elektronski pošti. Pri oddaji preko računalnika rešitev natipkaš neposredno v brskalniku. Med tipkanjem se rešitev na približno dve minuti samodejno shrani. Poleg tega lahko sam med pisanjem rešitve izrecno zahtevaš shranjevanje rešitve s pritiskom na gumb „Shrani spremembe“. Ker je vgrajeni urejevalnik dokaj preprost in ne omogoča označevanja kode z barvami, predlagamo, da rešitev pripraviš v urejevalniku na svojem računalniku in jo nato prekopiraš v okno spletnega urejevalnika. Naj te ne moti, da se bodo barvne oznake kode pri kopiranju izgubile.

Ko si bodisi zadovoljen z rešitvijo ter si zaključil nalogo ali ko želiš začasno prekiniti pisanje rešitve naloge ter se lotiti druge naloge, uporabi gumb „Shrani in zapri“ in nato klikni na „Nazaj na seznam nalog“, da se vrneš v glavni meni. (Oddano rešitev lahko kasneje še spreminjaš.) Za vsak slučaj priporočamo, da pred oddajo shraniš svoj odgovor tudi v datoteko na svojem lokalnem računalniku.

Med reševanjem lahko vprašanja za tekmovalno komisijo postavljaš prek zasebnih sporočil na tekmovalnem strežniku (ikona oblačka zgoraj desno), izjemoma pa tudi po elektronski pošti na `rtk-info@ijs.si`. Prek zasebnih sporočil bomo pošiljali tudi morebitna pojasnila in popravke, če bi se izkazalo, da so v besedilu nalog kakšne nejasnosti ali napake. Zato med reševanjem redno preverjaj, če so se pojavila kakšna nova zasebna sporočila.

Če imaš pri oddaji odgovorov prek spletnega strežnika kakšne težave, lahko izjemoma pošlješ svoje odgovore po elektronski pošti na `rtk-info@ijs.si`, vendar nas morajo doseči pred koncem tekmovanja; odgovorov, prejetih po koncu tekmovanja, ne bomo upoštevali.

Svoje odgovore dobro utemelji. Če pišeš izvorno kodo programa ali podprograma, **OBVEZNO** tudi v nekaj stavkih z besedami opiši idejo, na kateri temelji tvoja rešitev. Če ni v nalogi drugače napisano, lahko tvoje rešitve predpostavljajo, da so vhodni podatki brez napak (da ustrezajo formatu in omejitvam, kot jih podaja naloga). Zaželeno je, da so tvoje rešitve poleg tega, da so pravilne, tudi učinkovite; bolj učinkovite rešitve dobijo več točk (s tem je mišljeno predvsem, naj ima rešitev učinkovit algoritem; drobne tehnične optimizacije niso tako pomembne). **Nalog je pet** in pri vsaki nalogi lahko dobiš od 0 do 20 točk.

Rešitve bodo objavljene na `http://rtk.ijs.si/`. Predvidoma nekaj dni po tekmovanju bodo tam objavljeni tudi rezultati.

1. Gesla

Marjan je pozabil geslo za Apple ID in se ga ne spomni. Kot mnogi drugi ljudje ima tudi on nekaj različnih gesel, ki jih ponavadi uporablja za vse možne storitve (Gmail, Facebook, Instagram itd.). Ta gesla vsebujejo samo male črke angleške abecede in števke (na primer: „`iec4oovi`“, „`eipe9thu`“ in podobno); vsako geslo vsebuje vsaj eno črko. Marjan ni prepričan, katero geslo je sprva nameraval uporabiti za Apple ID, spomni pa se, da je geslo moralo biti „varno“, kar pomeni, da je moral Marjan v svojem geslu uporabiti tudi en znak, ki ni črka ali števka in pa vsaj eno veliko črko. Spomni se le še tega, da je nekje znotraj gesla (mogoče celo čisto na začetku ali na koncu) dodal piko in spremenil eno od obstoječih črk v veliko, ne spomni pa se natančno, kje je dodal piko in katero črko je spremenil v veliko. **Napiši podprogram** (funkcijo) MoznaGesla(geslo), ki kot parameter dobi niz geslo z Marjanovim prvotnim geslom iz samih malih črk in števk ter izpiše vse možne nize, ki bi lahko bili Marjanovo geslo za Apple ID. (Na primer: iz `eipe9thu` lahko dobimo `eip.E9thu` ali `e.ipe9tHu` ali `.eiPe9thu` ali še marsikaj drugega.)

2. Marsovci

Vsak marsovec se specializira za natanko 5 opravil. Če je za izvedbo naloge treba več kot 5 opravil, se povežejo v skupine. Imamo skupino m marsovcev in za vsakega marsovca imamo podatke o tem, katerih 5 opravil zna opravljati. Opravila so predstavljena s celimi števili od 1 do 100. **Napiši program**, ki za podano skupino marsovcev ugotovi, ali so vsa tista opravila, ki jih opravlja vsaj en marsovec v skupini, približno enako zastopana; natančneje povedano, preveriti moraš, ali se število marsovcev, ki so specializirani za posamezno opravilo, od enega opravila do drugega razlikuje največ za 1. Podatke naj tvoj program prebere s standardnega vhoda ali pa iz datoteke `marsovci.txt` (karkoli ti je lažje); v prvi vrstici je število marsovcev m, v vsaki od naslednjih m vrstic pa je po 5 števil, ki povedo, katera opravila obvlada posamezni marsovec. Če so vsa opravila približno enako zastopana, naj izpiše `da`, sicer pa `ne`.

Primer vhodnih podatkov:

```
4
75 12 96 57 28
96 28 12 75 9
96 9 57 28 75
12 57 9 28 75
```

Pripadajoči izhod:

```
da
```

Še en primer vhoda:

```
4
75 12 96 57 28
96 28 12 75 9
96 9 57 28 75
12 57 96 28 75
```

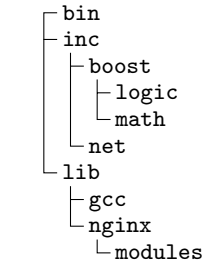
Pripadajoči izhod:

```
ne
```

Komentar: v prvem primeru se vsako opravilo pojavlja pri treh ali štirih marsovcih, zato so približno enakomerno zastopana. V drugem primeru pa se opravilo 9 pojavlja le pri dveh marsovcih, nekatera opravila pa pri štirih, zato niso približno enakomerno zastopana (glede na definicijo iz besedila naloge).

3. Rekonstrukcija poti

Direktorije oziroma mape na disku si pogosto predstavljamo kot zložene v drevesasto hierarhično strukturo, na primer takole:



Če bi hoteli takšno drevo direktorijev predstaviti samo z besedilom, brez črt, nam lahko prideta na misel naslednja dva načina:

(1) Pri vsakem imenu direktorija lahko zapišemo njegovo globino v drevesu. Pri zgornjem drevesu bi tako dobili:

```
bin 1
inc 1
boost 2
logic 3
math 3
net 2
lib 1
gcc 2
nginx 2
modules 3
```

(2) Lahko pa za vsak direktorij izpišemo polno pot od korena do njega. Pri zgornjem drevesu bi tako dobili:

```
/bin
/inc
/inc/boost
/inc/boost/logic
/inc/boost/math
/inc/net
/lib
/lib/gcc
/lib/nginx
/lib/nginx/modules
```

Napiši program, ki prebere predstavitev drevesa v prvi obliki (torej z imeni direktorijev in njihovimi globinami v drevesu) in ga izpiše v drugi obliki (torej s polnimi potmi). Delovati mora seveda za poljuben vhod, ne le za tistega iz gornjega primera. Če v vhodnem seznamu manjka kakšen direktorij in zato v nekem trenutku poti ni več mogoče rekonstruirati, naj program izpiše „`Napaka!`“ in se neha izvajati. Podatke lahko bereš s standardnega vhoda in pišeš na standardni izhod ali pa bereš iz datoteke `vhod.txt` in pišeš na `izhod.txt` (karkoli ti je lažje). Imena direktorijev so sestavljena le iz črk, brez presledkov ali kakšnih drugih posebnih znakov.

Primer vhoda, kjer rekonstrukcija ni mogoča:

```
abc 1
def 3
```

Tvoj program bi moral tu izpisati:

```
/abc
Napaka!
```

4. Kako dobri so virusni testi?

Prebivalce testiramo na okužbo z virusom covida s hitrimi testi in s testi PCR. Prvi so, kot že ime pove, hitri (in poceni) in dajo rezultat v nekaj minutah, so pa nezanesljivi, drugi, tako imenovani testi PCR, pa so zanesljivejši, vendar precej dražji in je na rezultate treba čakati en dan.

Da bi ugotovili kvaliteto hitrih testov, občasno testiramo skupino ljudi hkrati z obema vrstama testov, hitrimi in testi PCR. Rezultate lahko predstavimo z dvema enako dolgima nizoma znakov, pri čemer i-ti znak prvega niza pove rezultat hitrega testa na i-tem pacientu (`1` = okužen in `0` = neokužen), i-ti znak drugega niza pa rezultat testa PCR na istem pacientu (enako `1` = okužen in `0` = neokužen). Primerjava obeh nizov nam pokaže kvaliteto hitrih testov, ki smo jih pri tem poskusu uporabili.

Napiši podprogram (funkcijo) `Primerjava(s, t, n)`, ki kot parametra dobi dva enako dolga niza `s` (rezultati hitrih testov) in `t` (rezultati testov PCR) in ugotovi, pri katerih n zaporednih pacientih je bilo največ razhajanj med hitrimi in testi PCR. Tvoja funkcija naj vrne indeks, na katerem se začne ta skupina n zaporednih pacientov; če je takšnih skupin več, vrni indeks najbolj leve od njih (tiste z najmanjšim začetnim indeksom). Tvoja rešitev naj bo čim bolj učinkovita, da bo delovala hitro tudi za zelo dolge nize in velike n. Predpostavi, da sta `s` in `t` dolga po vsaj n znakov, tako da rešitev gotovo obstaja.

5. Zlaganje loncev

V kuhinjsko omaro zlagamo lonce. Lonci so v obliki odprtih valjev različnih premerov. Zaradi prihranka prostora lahko natanko en manjši lonec položimo v večjega, kadar ima manjši premer osnovne ploskve kot večji lonec. V ta manjši lonec pa lahko kasneje položimo še en manjši lonec in tako naprej, da dobimo nekakšen sklad loncev. Ne želimo pa v en lonec neposredno postaviti dveh ali več manjših (npr. da bi v lonec premera 20 cm postavili neposredno lonca premerov 5 cm in 3 cm; v tem primeru bi v lonec premera 20 cm postavili lonec premera 5 cm, v slednjega pa potem lonec premera 3 cm). Želimo preveriti, ali je naša kuhinjska omara dovolj prostorna, da lahko v njo na ta način postavimo vse svoje lonce.

Opiši postopek (ali napiši program ali podprogram oz. funkcijo, če ti je lažje), ki kot vhodni podatek dobi seznam premerov vseh loncev na kuhinjski mizi ter izračuna najmanjše število skladov, ki jih lahko sestavimo iz teh loncev. Izračuna pa naj tudi najnižjo možno vsoto, ki jo lahko dobimo, če vzamemo premer najbolj spodnjega lonca v vsakem skladu in te premere seštejemo po vseh skladih. Dobro tudi utemelji pravilnost svojega postopka.

Primer: če imamo lonce s premeri

$$28, 17, 14, 29, 12, 22, 28, 28, 13, 20, 30, 18, 4, 18, 4,$$

potrebujemo najmanj tri sklade, najmanjša možna vsota premerov pa je 86. (Eden od možnih načinov, kako lahko zložimo lonce na optimalen način, so takšni trije skladi: $[4, 14, 28, 29, 30]$, $[13, 17, 18, 28]$ in $[4, 12, 18, 20, 22, 28]$; vsota premerov najbolj spodnjih loncev je takrat $30 + 28 + 28 = 86$.)