15. tekmovanje ACM v znanju računalništva za srednješolce

28. marca 2020

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 tekmovalce diskvalificira, če bi se kasneje izkazalo, da nalog niso reševali sami. 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“. Gumb „Shrani in zapri“ uporabiš, ko si bodisi zadovoljen z rešitvijo ter si zaključil nalogo, ali ko želiš začasno prekiniti pisanje rešitve naloge ter se lotiti druge naloge. Po pritisku na ta gumb se vpisana rešitev shrani in te vrne v glavni menu. (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`.

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. Vesoljske vsote

Preučevalec vesoljske matematike, Tim, je na eni izmed svojih odprav v vesolje odkril skrivnosten tuj zapis vsot. Niz „`*---**-*-----*`“ predstavlja vesoljski zapis vsote $1+4+4+5+10$. Po podrobnem pregledu nekaj zapisov je ugotovil, da za zapisovanje vsot v vesolju obstajajo naslednja pravila:

- Na začetku postavimo trenutno število na 1.
- Znak „`*`“ doda trenutno število v zapis vsote (na konec).
- Znak „`-`“ poveča trenutno število za 1.

Tim ni najbolj spreten pri programiranju, zato te prosi, da mu **napišeš program** (ali podprogram oz. funkcijo), ki vesoljski zapis vsote pretvori v človeku berljiv račun. Na standardni vhod bo tvoj program dobil niz znakov „`*`“ in „`-`“, ki naj ga pretvori v človeku berljiv račun (na koncu naj doda tudi enačaj in končno vrednost vsote) in izpiše na standardni izhod (ali pa v datoteko `vsota.txt`, karkoli ti je lažje). Predpostaviš lahko, da vsebuje vhodni niz vsaj eno zvezdico „`*`“. Zgornji niz naj tvoj program izpiše kot „`1 + 4 + 4 + 5 + 10 = 24`“.

Tim je pripravil tudi dva primera, da mu boš lažje pomagal.

Primer 1:

Vhod: `*---**-*-----*`
Izhod: `1 + 4 + 4 + 5 + 10 = 24`

Naslednja tabela kaže vrednost trenutnega števila po vsakem prebranem znaku:

Znak		*	-	-	-	*	*	-	*	-	-	-	-	-	*
Število	1	1	2	3	4	4	4	5	5	6	7	8	9	10	10

Primer 2:

Vhod: `-----*-*`
Izhod: `6 + 7 = 13`

Naslednja tabela kaže vrednost trenutnega števila po vsakem prebranem znaku:

Znak		-	-	-	-	-	*	-	*
Število	1	2	3	4	5	6	6	7	7

2. Ključ

Ključ za običajno cilindrično ključavnico ima šest zarez, globina vsake se mora ujemati z dolžino istoležnega zatiča v ključavnici, da se ključavnica lahko odklene. Zareze niso poljubno globoke, ampak proizvajalec predpisuje določene pogoje, ki jih morajo globine zarez izpolnjevati, da lahko nek tip ključavnice deluje pravilno in zanesljivo ter da se lahko ključ vanjo vstavi in izvleče brez zatikanja. Možnih je deset različnih globin, označenih s številko med 0 in 9.

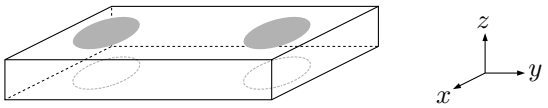
1. dovoljene so le globine med 1 in 8;
2. globini sosednjih dveh zarez se lahko razlikujeta za največ 5 (globina 2 na primer ne sme biti sosednja globini 8; to pravilo zagotavlja, da se ključ ne more zatakniti pri vstavljanju ali odstranjevanju);
3. dve sosednji zarezi ne smeta imeti enake globine;
4. nobena globina zareze se ne sme pojaviti več kot dvakrat.

Napiši podprogram (funkcijo), ki bo kot argument prejel celoštevilčno kodo ključa (na primer: 597969). Desetiške števke tega števila predstavljajo šest globin zarez. Lahko je prvih nekaj števk enakih 0 (na primer: število 123 predstavlja kodo 000123). Za vsako od naštetih štirih pravil naj podprogram izpiše, ali je pravilo izpolnjeno ali ne.

Primer: 597969 ustreza drugemu in tretjemu pogoju, ne pa prvemu (ker v njem niso samo števke od 1 do 8) in četrtemu (ker se števka 9 pojavlja trikrat).

3. Obračanje jogija

Imamo jogi v obliki kvadra. Odvisno od tega, kako ga obrnemo, se lahko naša glava znajde na enem od štirih možnih mest, kot kaže leva slika spodaj:



Sivo pobarvani elipsi kažeta dva možna položaja glave na eni strani jogija, črtkani elipsi pa še dva možna položaja glave na drugi strani jogija.

Spati želimo tako, da imamo glavo na najmanj obrabljenem delu, zato smo pripravljeni jogi občasno obrniti — zavrtimo ga za 180 stopinj okrog ene od osi x, y ali z (desna slika zgoraj kaže, kam je usmerjena katera os), tako da bo potem naša glava na kakšnem od ostalih treh možnih mest na jogiju.

Napiši funkcijo ObrniJogi(n), ki jo bo uporabnik poklical, ko bo pripravljen obrniti jogi, ona pa mu mora primerno svetovati, po kateri osi naj obrne jogi za 180 stopinj, da bo imel glavo na najmanj obrabljenem delu (torej na tistem, na katerem je doslej spal najmanj dni). Funkcija naj vrne enega od znakov `'x'`, `'y'`, `'z'`; če pa je bolje, da uporabnik trenutno jogija sploh ne obrača, naj tvoja funkcija vrne znak `'n'`. Uporabnik potem obrne jogi v skladu z rezultatom, ki ga vrne tvoja funkcija, in vse odtlej do naslednjega klica vsak dan spi na njem v njegovem sedanjem položaju (jogija torej nikoli ne obrača na lastno pest).

Kot parameter `n` bo tvoja funkcija dobila število dni od zadnjega klica (toliko dni je uporabnik spal na dosedanjem mestu).

Predpostavi, da uporabnik tvojo funkcijo prvič pokliče z `n = 0` in da dotlej na jogiju ni še nikoli spal. Poleg funkcije ObrniJogi lahko deklariraš tudi poljubne globalne spremenljivke (oz. spremenljivke zunaj svoje funkcije) in jih po želji inicializiraš.

4. Zobna ščetka

Električno zobno ščetko poganja elektromotor, kot uporabniški vmesnik pa služi tipka; ta je lahko pritisnjena ali spuščena. Delovanje motorja upravlja preprost računalnik, ki lahko odčitava trenutno stanje tipke in lahko vklaplja ali izklaplja motor. Izklopljeno ščetko spravimo v pogon s pritiskom na tipko. Trajanje pritiska na tipko ne vpliva na delovanje, važen je le trenutek začetka pritiska tipke. Da uporabnik ne pretirava s čiščenjem zob, se mora ščetka samodejno izklopiti po 120 sekundah od zadnjega vklopa, lahko pa jo uporabnik izklopi že pred iztekom tega časa s (ponovnim) pritiskom na tipko.

Napiši program, ki bo upravljal z motorjem zobne ščetke tako, kot je zgoraj predpisano. Na razpolago so naslednje funkcije:

- za odčitavanje stanja tipke:

  Tipka() — funkcija vrne **true**, če je tipka pritisnjena, sicer **false**. (Funkcija ne čaka na pritisk tipke, ampak se vrne takoj in sporoči trenutno stanje tipke. Če uporabnik dlje časa drži tipko pritisnjeno, funkcija v tem času ob vsakem klicu vrne **true**.)

- za vklop ali izklop motorja:

  Motor(vklop) — če ima parameter vklop vrednost **true**, bo funkcija vklopila motor, če ima vrednost **false**, pa ga bo izklopila.

- štoparica, ki meri čas v sekundah:

  PozeniUro() — postavi čas na 0 in požene štoparico;
  UstaviUro() — ustavi štoparico;
  OdcitajUro() — vrne čas štoparice v sekundah kot celo število (tipa **int** oz. integer).

(Kdor piše v pythonu, naj si namesto **true** in **false** misli True in False.)

5. Plonkanje

Zaradi pojava virusa so morali organizatorji nekega tekmovanja iz znanja izvesti le-to prek interneta, kjer pa tekmovalna komisija ni mogla nadzirati, če so si tekmovalci med seboj kaj pomagali („plonkali“).

Po natančni analizi vseh oddanih nalog več sto tekmovalcev je tekmovalni komisiji uspelo (seveda s pomočjo računalnika) natančno rekonstruirati, kdo je plonkal od koga.

Podatki o plonkanju so (zaradi varstva osebnih podatkov) anonimizirani in predstavljeni v tabeli z dvema stolpcema, kjer prva številka pomeni številko tekmovalca, ki je bil *pomočnik*, druga pa številko *prepisovalca*, torej tekmovalca, ki je prepisoval (plonkal). Tisti, ki je plonkal, je seveda lahko bil ob neki drugi priložnosti pomočnik in je pomagal novemu prepisovalcu in tako naprej. Pomočnik je lahko pomagal več prepisovalcem (v prvem stolpcu bodo lahko tudi enake številke), medtem ko je prepisovalec vedno lahko plonkal samo od enega pomočnika (v drugem stolpcu bodo same različne številke). Primer:

pomočnik	prepisovalec (plonkar)
15	18
41	62
15	29
47	50
29	47
15	41
33	21
91	55
41	37
21	12
12	72
12	33

Iz zgornje tabele ugotovimo na primer, da je tekmovalec 41 prepisoval od tekmovalca 15 in da je tekmovalec 41 pomagal tekmovalcu 62 in tekmovalcu 37. Tekmovalec 15 pa je pomagal tudi tekmovalcu 29 in tekmovalcu 18.

Opiši postopek, ki bo iz tako podanih podatkov ugotovil:

(a) kateri so bili tisti tekmovalci, ki niso plonkali od nikogar, ampak so bili izključno pomočniki — imenujmo jih *izvirni tekmovalci*;

(b) kateri so bili *plonkarji prvega reda*, torej tekmovalci, ki so plonkali od izvirnih tekmovalcev;

(c) kateri so bili *plonkarji drugega reda*, torej tekmovalci, ki so plonkali od nekoga, ki je sam plonkal od izvirnega tekmovalca.

V zgornjem primeru sta izvirna tekmovalca 15 in 91, plonkarji prvega reda so 18, 41, 29 in 55, plonkarji drugega reda pa so 37, 62 in 47.