15. tekmovanje ACM v znanju računalništva za srednješolce

28. marca 2020

NALOGE ZA DRUGO 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. Metanje na koš

Profesionalni košarkarji se morajo med treningom vaditi tudi v metanju prostih metov na koš, saj so prosti meti pomemben del košarkaških tekem in je važno, da so pri tem čim bolj zanesljivi.

Organiziramo torej tekmovanje v metanju prostih metov na koš, kjer se tekmovalci preizkušajo, kdo ima najboljše živce in bo največkrat vrgel na koš, ne da bi enkrat samkrat zgrešil koš.

Pravila so takšna: Vsak tekmovalec bo žogo na koš vrgel velikokrat, recimo tisočkrat (seveda lahko vmes tudi počiva, spije vodo, ali kaj prigrizne `;-)`). Zmagovalec tekmovanja bo tisti, ki bo naredil najdaljši neprekinjeni niz zadetkov v koš.

Če pa bo imelo več tekmovalcev enak najdaljši neprekinjeni niz zadetkov, potem bo zmagovalec tisti od njih, ki ima najdaljši niz zadetkov, v katerem je en sam met mimo koša.

In če bo tudi pri tem pogoju še vedno več tekmovalcev imelo enako dolg niz zadetkov z enim zgrešenim metom, potem bo zmagovalec tisti, ki bo imel najdaljši niz zadetkov z dvema zgrešenima metoma. In tako dalje, s tremi, štirimi, petimi zgrešenimi meti v nizu, dokler ne bo število zgrešenih metov v zaporednem nizu enako številu vseh zgrešenih metov tekmovalca. Če bo tudi takrat več tekmovalcev imelo enak rezultat (pri tisoč metih je sicer zelo malo verjetno, da bi do tega prišlo), bo zmagovalca pač določila tekmovalna komisija z žrebom.

Podatki za enega tekmovalca so predstavljeni z nizom enic in ničel (enica za zadetek v koš, ničla za met mimo koša), na primer takole (35 metov, 26 zadetkov, 9 mimo koša):

```
11101111001101111101111001011111110
```

Opiši postopek ali napiši podprogram (funkcijo), ki za dani niz in celo število k izračuna dolžino najdaljšega takega niza zadetkov, med katerimi je največ k metov mimo koša. (Če tega ne znaš rešiti v splošnem, reši nalogo vsaj za primere, ko je $k = 0$ ali $k = 1$, in boš dobil delne točke.)

Primer: za gornji niz 35 metov bi pri $k = 0$ dobili rezultat 7, pri $k = 1$ rezultat 9 (zaporedje petih zadetkov, nato en met mimo koša in nato še zaporedje štirih zadetkov), pri $k = 3$ pa rezultat 12.

2. Ne odlašaj na jutri, kar lahko storiš pojutrišnjem

Učenci v šoli imajo n obveznosti, ki jih morajo izpolniti. Pri tem jim i-ta obveznost vzame d_i dni in jo morajo opraviti najkasneje na dan k_i (mislimo si, da so dnevi oštevilčeni z naravnimi števili od začetka šolskega leta). V posameznem dnevu se lahko ukvarjajo samo z eno obveznostjo, lahko pa počivajo in se ne ukvarjajo z nobeno. Ko se enkrat lotijo neke obveznosti, jo morajo potem opraviti v neprekinjenem sklopu d_i zaporednih dni (torej jim ta obveznost vzame dan, ko so začeli z njo, in še naslednjih $d_i - 1$ dni). Če je na primer $d_i = 3$, to pomeni, da se morajo začeti z i-to obveznostjo ukvarjati najkasneje na dan $k_i - 2$.

Kot tipični učenci si seveda želijo vse te obveznosti opraviti čim kasneje. Torej, če lahko nek dan počivajo in se obveznosti lotijo jutri in še zmeraj opravijo vse obveznosti pravočasno, potem danes seveda raje počivajo.

Natančneje povedano, dva možna razporeda tega, kdaj opravijo kakšno obveznost, primerjamo takole: poiščemo najzgodnejši dan, na katerega pri enem razporedu počivajo, pri drugem pa delajo; če takega dne ni, štejemo razporeda za enakovredna in sta nam oba enako dobra; sicer pa nam je boljši tisti razpored, pri katerem na tisti dan počivajo.

Napiši program ali podprogram (funkcijo), ki kot vhodne podatke dobi podatke o obveznostih (števila $n, k_1, d_1, k_2, d_2, \ldots, k_n, d_n$) in izračuna najboljši možni razpored (za vsako obveznost i naj torej ugotovi, na kateri dan z_i se morajo začeti ukvarjati z njo), ali pa ugotovi, da za te vhodne podatke sploh ni nobenega veljavnega razporeda. Če je možnih več enako dobrih razporedov, je vseeno, katerega poiščeš. Podrobnosti glede oblike vhodnih in izhodnih podatkov si izberi sam in jih tudi opiši. Tvoja rešitev naj bo čim učinkovitejša, tako da bo uporabna tudi za večje vhodne primere (recimo, da gre lahko n do 10^6, datum k_i pa do 10^9).

3. Lenoba

V službi želimo preživeti čim manj časa, ne da bi to kdorkoli opazil. Za vse sodelavce natanko vemo čas prihoda in odhoda (vsakdo pride in odide zgolj enkrat v dnevu; vsi podatki so znotraj enega dneva, nihče ne ostane v službi čez polnoč). **Opiši postopek**, ki izračuna, kdaj moramo priti v službo in koliko časa moramo tam preživeti, da nobena oseba ne bo prisotna takrat, ko pridemo, in še vedno prisotna takrat, ko odidemo. Edini dodatni pogoj je, da želimo priti v službo pred dvanajsto in oditi po dvanajsti (ker je točno ob dvanajstih kosilo).

Kot vhodne podatke tvoj postopek dobi število sodelavcev in za vsakega sodelavca čas njegovega prihoda in odhoda. Vsi časi se merijo v nanosekundah od polnoči, tako da so to sicer nenegativna cela števila, vendar so lahko precej velika. (Ena sekunda ima 1 000 000 000 nanosekund.) Sodelavec, ki pride ali odide ob istem času kot mi, nas vidi priti ali oditi — če se hočemo temu izogniti, moramo priti vsaj eno nanosekundo pred njim ali oditi vsaj eno nanosekundo za njim. Podrobnosti glede predstavitve vhodnih podatkov si izberi sam in jih v svoji rešitvi tudi opiši.

4. Semafor

Semafor na prehodu za pešce ima dvomestni prikazovalnik sekund do spremembe luči. Prikazuje lahko torej poljubno celo število od 0 do 99, lahko pa je tudi ugasnjen. **Napiši podprogram** oz. funkcijo VsakoSekundo(n), ki jo bo sistem poklical enkrat na sekundo, da mu bo pomagala upravljati s prikazovalnikom na semaforju. Prek parametra n ti sistem pove, kaj se dogaja z lučjo semaforja: če se je v zadnji sekundi stanje luči spremenilo (iz zelene v rdečo ali obratno), ti parameter n pove, koliko sekund bo luč semaforja v svojem novem stanju; če pa se v zadnji sekundi stanje luči ni spremenilo, boš dobil $n = -1$.

Na voljo imaš funkcijo Prikazi(n), ki jo lahko pokličeš, da na prikazovalniku prikažeš število n. Število n mora biti od 0 do 99, lahko pa je -1, če hočeš, da naj bo prikazovalnik prazen (ne prikazuje nobenega števila, niti ničle).

Deklariraš lahko tudi globalne spremenljivke in jih po svoji želji inicializiraš. Predpostavi, da bo vrednost parametra n pri prvem klicu funkcije VsakoSekundo gotovo večja od 0.

Če čas, ko bi se morala luč spremeniti, mine, sistem pa te še ni obvestil o spremembi stanja luči (in trajanju novega stanja), moraš poskrbeti, da bo prikazovalnik prazen, dokler te sistem ne obvesti o spremembi stanja luči.

Prikazovalnik na semaforju je le dvomesten, stanje luči pa včasih traja več kot 99 sekund. Zato naj tvoj podprogram poskrbi, da če je do spremembe stanja luči več kot 99 sekund, naj se prikazuje število 99, vendar naj utripa (eno sekundo gori, eno sekundo je ugasnjena).

5. Prelom besedila

Dano imamo besedilo, dolgo z znakov, ki ga želimo prikazati v okencu, ki ima prostora za w znakov na vrstico. Ko besedilo pišemo v okence, lahko vrstico prelomimo le na presledku pred začetkom nove besede. Presledki ostanejo na prejšnji vrstici in lahko gledajo preko roba okna, nova vrstica pa se začne s prvim naslednjim znakom, ki ni presledek.

Za vsako širino w od 1 do z izračunaj, koliko vrstic bo besedilo imelo, če ga želimo napisati v okence širine w. Besedilo bo vsebovalo le črke, številke, ločila in presledke. Lahko predpostaviš, da drugih znakov za prazen prostor, kot na primer tabulatorjev ali znakov za novo vrstico, ne bo. Besedilo se tudi ne bo začelo s presledkom.

Napiši podprogram (funkcijo), ki sprejme besedilo kot niz znakov dolžine z in vrne ali izpiše seznam z števil, kjer števila po vrsti predstavljajo, koliko vrstic bo besedilo imelo, če ga želimo napisati v okence širine $1, 2, 3, 4, \ldots, z$. Če besedila v okence neke širine ni mogoče spraviti, ne da kakšna beseda gledala prek meja, na tisto mesto napiši -1.

Primer. Recimo, da dobimo takšen vhodni niz (spodaj je napisan v dveh vrsticah, vendar si moramo obe skupaj predstavljati kot en sam niz, brez kakšnih vmesnih znakov za konec vrstice ali česa podobnega; presledki so predstavljeni s simbolom ␣, da se jih bolje vidi):

```
Na␣začetku␣je␣bilo␣ustvarjeno␣vesolje.␣␣To␣je␣povzročilo␣
mnogo␣hude␣krvi␣in␣na␣splošno␣velja␣za␣zelo␣slabo␣potezo.
```

Pravilen izhodni seznam za ta niz je:

$-1, -1, -1, -1, -1, -1, -1, -1, -1, 12, 12, 12, 10, 10, 8, 8, 8, 7, 6, 6, 6,$
$6, 6, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,$
$3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,$
$2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,$
$2, 2, 1.$

Na primer, če je $w = 19$, spravimo besedilo v 6 vrstic (zato je 19. element v gornjem seznamu enak 6):

```
Na␣začetku␣je␣bilo␣
ustvarjeno␣vesolje.␣␣
To␣je␣povzročilo␣
mnogo␣hude␣krvi␣in␣
na␣splošno␣velja␣za␣
zelo␣slabo␣potezo.
```

(Primer očitno povzet po: Douglas Adams, *Restavracija ob koncu Vesolja*, *Štoparski vodnik po galaksiji*, prevod: Alojz Kodre.)