Drugi izpitni rok pri predmetu Programiranje 1
31. januar 2022

Oddajte datoteke `Prva.java`, `Druga.java`, `Tretja.java` in `Cetrta.java`. Testirate jih lahko takole:

(1) `tj.exe Prva.java . .` (2) `tj.exe` (3) `tj.exe` (4) `tj.exe`

(1) Branjevke na tržnici prodajajo solato. Na začetku je imajo vse po $s \in [1, 10^4]$ glav. Kupci se ne obnašajo najbolj pošteno: solato kupujejo izključno pri prvi branjevki, a le tako dolgo, dokler eden od njih ne zahteva več glav, kot jih ima branjevka na voljo. Ta kupec se nemudoma preseli k drugi branjevki, ne da bi pri prvi karkoli kupil. Vsi naslednji kupci kupujejo pri drugi branjevki, dokler solate ne zmanjka še tej. Nato se kupci preselijo k tretji branjevki itd.

Napišite program (`Prva.java`), ki prebere števili s in $k \in [1, 10^4]$ (število kupcev) ter k števil z intervala $[1, s]$, ki podajajo zahteve posameznih kupcev (število glav, ki jih želijo kupiti), izpiše pa zaporedno številko branjevke, ki postreže zadnjega (k-tega) kupca.

Primer (vhod/izhod):

```
10 7
3
5
4
7
1
2
9
```

```
4
```

Branjevka 1 postreže prva dva kupca, branjevka 2 tretjega, branjevka 3 četrtega, petega in šestega, branjevka 4 pa sedmega (zadnjega).

(2) Slika višine h in širine w je predstavljena s tabelo `slika` velikosti $h \times w \times 3$. Element `slika[`i`][`j`][`k`]` podaja vsebnost barvne komponente k (0: rdeča; 1: zelena; 2: modra) za piko (»piksel«) v vrstici z indeksom i in stolpcu z indeksom j.

Naj bo pika *prevladujoče zelena*, če ima zelene barvne komponente strogo več od ostalih dveh. Na primer, pika z barvnimi komponentami (70, 90, 80) je prevladujoče zelena, pika (70, 90, 90) pa ni.

V razredu `Druga` napišite metodo

`public static int stolpecZNajvecPrevladujoceZelenimi(int[][][] slika)`,

ki vrne indeks stolpca, ki vsebuje največ prevladujoče zelenih pik. Lahko predpostavite, da je tak stolpec en sam.

V 50% testnih primerov velja, da slika vsebuje eno samo prevladujoče zeleno piko.

(3) Predavalnice v fakultetni stavbi so treh tipov: avditorne predavalnice, računalnice, v izrednih razmerah pa lahko to vlogo prevzamejo celo garaže. Razred `Tretja` vsebuje pripadajoče statične notranje razrede:

```
abstract class Predavalnica {
    private String oznaka;     // npr. PR15
    private int stMest;        // število delovnih mest
}
class Avditorna extends Predavalnica {}
class Racunalnica extends Predavalnica {
    private int stRacunalnikov;   // ⩽ stMest
}
class Garaza extends Predavalnica {
    private int povrsina;    // površina v m^2
}
class Stavba {
    private Predavalnica[] predavalnice;    // vse predavalnice v stavbi
}
```

Rešite sledeči nalogi:

- [50%] V razred `Predavalnica` in po potrebi v njegove podrazrede dodajte metodo

  `public int casCiscenja()`,

  ki vrne čas čiščenja predavalnice. Za vsako delovno mesto sta potrebni dve časovni enoti, za vsak računalnik (še) tri enote, za vsak kvadratni meter površine garaže pa (še) ena enota. Na primer, za čiščenje garaže s 100 delovnimi mesti in površino 500 m^2 potrebujemo $100 \cdot 2 + 500 \cdot 1 = 700$ časovnih enot.

- [50%] Podano število študentov (`stStudentov`) razporedimo po predavalnicah stavbe, pri čemer najprej zapolnimo vse avditorne predavalnice (v vrstnem redu, kot so nanizane v tabeli `predavalnice`), nato na enak način vse računalnice, nazadnje pa na enak način še garaže. V razred `Stavba` dodajte metodo

  `public int[] razporedi(int stStudentov, int[] ostanek)`,

  ki vrne tabelo, v kateri i-ti element podaja končno zasedenost i-te predavalnice v tabeli `predavalnice`, v celico tabele `ostanek` z indeksom 0 pa vpiše število nerazporejenih študentov (to število bo seveda večje od 0 le v primeru, če bodo vse predavalnice zasedene).

(4) V statičnem notranjem razredu `Tocka` z atributoma `x` in `y` (tipa `int`) napišite sledeči metodi:

- [50%] `public Map<Boolean, List<Tocka>> razdeli(Collection<Tocka> tocke)`

  Vrne slovar, v katerem se ključ `true` preslika v seznam tistih točk iz podane zbirke, ki imajo večjo koordinato x kot točka `this`, ključ `false` pa v seznam vseh ostalih točk.

- [50%] `public static Comparator<Tocka> polarno()`

  Vrne primerjalnik, ki podani točki primarno primerja po oddaljenosti od izhodišča $O = (0, 0)$ (točke bliže izhodišču sodijo pred tiste bolj oddaljene), sekundarno pa po polarnem kotu (točke z manjšim polarnim kotom sodijo pred tiste z večjim). Polarni kot točke T je kot med vektorjema $\overrightarrow{OE}$ in $\overrightarrow{OT}$, kjer je $E = (1, 0)$. Na primer, polarni kot točke $(1, 1)$ znaša $\frac{\pi}{4}$, polarni kot točke $(1, -1)$ pa $\frac{7\pi}{4}$.

  V vseh testnih primerih za vse točke velja $x \neq 0$ in $y \neq 0$.