Prvi izpitni rok pri predmetu Programiranje 1
31. januar 2024

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

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

Vsa števila, ki nastopajo v navodilih in testnih primerih tega izpita, so cela.

(1) Pravokotnik naj bo *skoraj-kvadrat*, če sta dolžini njegovih stranic celi števili in če je dolžina njegove krajše stranice strogo večja od polovice dolžine njegove daljše stranice. Na vhodu je podano število $n \in [1, 10^3]$ in zaporedje n števil z intervala $[2, 10^4]$, vaš program pa naj izpiše n vrstic: v i-ti vrstici (za $i \in \{1, \ldots, n\}$) naj izpiše `DA`, če je i-to število zaporedja lahko ploščina skoraj-kvadrata, in `NE`, če to ni mogoče.

V primeru na desni je število 45 ploščina skoraj-kvadrata 5×9. Števili 36 in 24 sta ploščini skoraj-kvadratov 6×6 oziroma 4×6.

Vhod:

```
5
32
45
21
36
24
```

Izhod:

```
NE
DA
NE
DA
DA
```

(2) Na dražbi se $m \in [1, 10^3]$ dražiteljev poteguje za $n \in [1, 10^3]$ izdelkov. Tako dražitelji kot izdelki so oštevilčeni z indeksi od 0 naprej. V vsaki ponudbi eden od dražiteljev za enega od izdelkov ponudi določen znesek (število z intervala $[1, 10^9]$). Če ponudba ne presega doslej najvišje ponudbe za tisti izdelek, se šteje kot neveljavna in se ignorira. Vsak izdelek dobi tisti dražitelj, ki zanj ponudi največ.

V prvi vrstici vhoda so podana števila $u \in \{1, 2\}$ (ukaz), m, n in $k \in [1, 10^3]$ (število ponudb). V vsaki od naslednjih k vrstic so podana tri števila: indeks dražitelja, indeks izdelka in ponujeni znesek.

Če je $u = 1$, naj program za vsak izdelek izpiše ceno, po kateri je bil prodan. (Če za nek izdelek ni bilo nobene ponudbe, naj program zanj izpiše 0.) Če je $u = 2$, naj program za vsakega dražitelja izpiše število izdelkov, ki jih je dobil. V obeh primerih naj bo izpis v obliki `[`t_0`, `t_1`, ...]` (pomagajte si z metodo `Arrays.toString`).

Lastnosti testnih primerov: [50%] $u = 1$; [50%] $u = 2$.

Primer 1 (vhod/izhod):

```
1 5 4 7
3 0 100
0 2 400
2 0 200
0 2 300
3 3 150
2 2 400
2 3 250
```

```
[200, 0, 400, 250]
```

Primer 2 (vhod/izhod):

```
2 5 4 7
3 0 100
0 2 400
2 0 200
0 2 300
3 3 150
2 2 400
2 3 250
```

```
[1, 0, 2, 0, 0]
```

Dražitelj 0 je kupil izdelek 2 po ceni 400 (njegova druga ponudba za isti izdelek se ignorira, neveljavna pa je tudi ponudba dražitelja 2 za ta izdelek). Dražitelj 2 je kupil izdelek 0 po ceni 200 in izdelek 3 po ceni 250.

(3) V podjetju Totalni nadzor, d.o.o., vestno beležijo izhode svojih zaposlenih. Izhodi so treh vrst: malica, rekreacija in službena pot. Vsak izhod zabeležijo kot dva objekta ustreznega podtipa tipa `Dogodek`: kot odhod (takrat ima atribut `odhod` vrednost `true`) in kot vrnitev na delo (takrat ima atribut `odhod` vrednost `false`). Ob vsakem dogodku se zabeleži tudi indeks osebe, ki je odšla oziroma prišla, in trenutek, ko se dogodek zgodi (objekt razreda `Cas`). Indeksi oseb se pričnejo z 0.

Objekt razreda `Dnevnik` hrani število zaposlenih (`stZaposlenih`) in kronološko urejeno tabelo vseh dogodkov. V tem razredu dopolnite sledeče metode:

- [32%] `public static int steviloMalicarjev()`

  Vrne število oseb, ki so šle na malico. Lahko predpostavite, da nihče ni šel na malico več kot enkrat, vsekakor pa upoštevajte, da se vsak izhod na malico (tako kot ostali izhodi) zabeleži dvakrat: kot odhod in kot vrnitev.

- [34%] `public static int kolikoRekreacije(int oseba)`

  Vrne skupno trajanje rekreacije (v minutah), ki si jo je privoščila oseba s podanim indeksom. Oseba se lahko rekreira tudi večkrat na dan.

- [34%] `public static boolean[] prisotnost(Cas cas)`

  Vrne tabelo s `stZaposlenih` elementi, pri čemer ima element z indeksom i vrednost `true` natanko v primeru, če je oseba z indeksom i ob času `cas` prisotna na delovnem mestu. Če oseba odide ob času t_1 in se vrne ob času t_2, potem ob času t_1 ni več prisotna, ob času t_2 pa je že prisotna.

(4) Dopolnite metodo

```
public static <T extends Comparable<T>>
    Iterator<T> zlitje(Iterator<T> a, Iterator<T> b),
```

ki vrne iterator, ki v naravnem vrstnem redu obišče vse objekte, ki jih obiščeta iteratorja `a` in `b`, če veste, da iteratorja `a` in `b` prav tako obiskujeta objekte v naravnem vrstnem redu. Na primer, če iterator `a` obišče nize `Ana`, `Cvetka` in `Franci` (to pomeni, da njegova metoda `next` najprej vrne `Ana`, potem `Cvetka`, potem `Franci`, nato pa `hasNext` vrne `false`), iterator `b` pa nize `Bojan`, `Denis` in `Eva`, potem mora izhodni iterator obiskati nize `Ana`, `Bojan`, `Cvetka`, `Denis`, `Eva` in `Franci` (v tem vrstnem redu).

Lastnosti testnih primerov (naj bodo $a_0 < a_1 < \ldots < a_{m-1}$ objekti, ki jih obišče iterator `a`, $b_0 < b_1 < \ldots < b_{n-1}$ pa objekti, ki jih obišče iterator `b`):

- [20%] $m = n = 1$, $a_0 < b_0$.
- [20%] $m = n \geqslant 1$, $a_{m-1} < b_0$.
- [20%] $m = n \geqslant 1$, $a_0 < b_0 < a_1 < b_1 < a_2 < b_2 < \ldots < a_{m-1} < b_{n-1}$.
- [40%] $m \geqslant 1$, $n \geqslant 1$.

POZOR! V svoji rešitvi lahko uporabljate izključno primitivne tipe, tipni parameter `T` in tip `Iterator<T>`. Uporaba ostalih tipov (tabele, `String`, `Collection` in njegovi podtipi, `Collections` itd.) je prepovedana in vam bo pri tej nalogi prinesla 0 točk ne glede na delovanje metode `zlitje` na skritih testnih primerih.