# Drugi izpitni rok pri predmetu Programiranje 1
# 10. februar 2021

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`

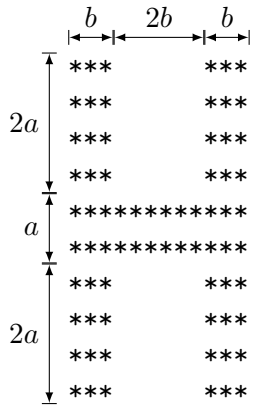
**(1)** Napišite program, ki prebere števili $a \in [1, 50]$ in $b \in [1, 50]$ in s pomočjo zvezdic in presledkov proizvede izpis po vzoru sledečega primera. Program naj ne izpiše nobenega odvečnega presledka.

V 50% skritih testnih primerov velja $a = b$.

Sledi primer vhoda in pripadajočega izhoda ter obrazložitev:

```
2 3
```

```
***         ***
***         ***
***         ***
***         ***
**********
**********
***         ***
***         ***
***         ***
***         ***
```




**(2)** Gorovje predstavimo z dvojiško matriko, v kateri ničle predstavljajo nebo, enice pa zemljo. Vsak stolpec matrike je sestavljen iz (lahko tudi praznega) zaporedja ničel in nepraznega zaporedja enic.

Alpinist prične svoj pohod na prvem vrhu in ga zaključi na zadnjem. Napišite program, ki izpiše skupno dolžino njegovega vzpona in skupno dolžino njegovega spusta.

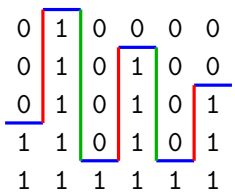
V prvi vrstici sta podani števili $h \in [1, 100]$ in $w \in [1, 100]$, nato pa sledi zapis matrike — $h$ vrstic s po $w$ ničlami in enicami. Števila v isti vrstici so med seboj ločena s po enim presledkom. Program naj izpiše dve vrstici: prva naj vsebuje dolžino vzpona, druga pa dolžino spusta.

V 50% skritih testnih primerov je skupna dolžina spusta enaka 0.

Sledi primer vhoda in pripadajočega izhoda ter obrazložitev:

```
5 6
0 1 0 0 0 0
0 1 0 1 0 0
0 1 0 1 0 1
1 1 0 1 0 1
1 1 1 1 1 1
```

```
8
7
```



Vzpon: $3 + 3 + 2 = 8$
Spust: $4 + 3 = 7$

(3) Podani so sledeči razredi (kot statični notranji razredi v razredu `Tretja`):

```
class Oseba {      // turist(-ka)
    private String ip;       // ime in priimek, npr. Ana Arko
    private String drzava;   // država izvora, npr. Slovenija
}
class Cilj {       // turistični cilj
    private String kraj;     // npr. Dunaj
    private String drzava;   // npr. Avstrija
}
class Nocitev {    // podatki o posamezni nočitvi
    private Oseba oseba;     // kdo je opravil nočitev
    private Cilj cilj;       // kje je bila opravljena nočitev
    private int leto;        // leto, v katerem je bila opravljena nočitev
}
```

Zunanji razred (`Tretja`) dopolnite s sledečimi metodami:

- [34%] `public static int notranje(Nocitev[] nocitve)`

  Vrne število nočitev iz tabele `nocitve`, pri katerih je turist(-ka) prenočeval(-a) v svoji lastni državi.

- [32%] `public static boolean jeZvesta(Nocitev[] nocitve, Oseba oseba)`

  Vrne `true` natanko v primeru, če je podana oseba glede na podatke v tabeli `nocitve` vseskozi prenočevala na enem in istem cilju (in nikoli nikjer drugje, niti znotraj iste države). Če oseba ni opravila nobene nočitve, je pravilen rezultat seveda `true`.

- [34%] `public static int[][] obiskanost(Nocitev[] nocitve, Cilj[] cilji, int minLeto, int maxLeto)`

  Izdela in vrne tabelo velikosti $C \times L$, kjer je $C$ dolžina tabele `cilji`, $L$ pa dolžina zaprtega intervala [`minLeto`, `maxLeto`]. Element vrnjene tabele na vrstičnem indeksu `i` in stolpčnem indeksu `j` naj podaja število nočitev iz tabele `nocitve`, ki so bile opravljene na cilju `cilji[i]` v letu `minLeto + j`.

(4) Napišite sledeči metodi:

- [60%] `public static <T> List<T> razmnozi(List<T> seznam, int n)`

  Izdela in vrne seznam, sestavljen iz ene kopije prvega elementa podanega seznama, dveh kopij drugega elementa, ..., $n$ kopij $n$-tega elementa, ene kopije elementa z zaporedno številko $(n+1)$, dveh kopij elementa z zaporedno številko $(n+2)$, ..., $n$ kopij $(2n)$-tega elementa, ene kopije elementa z zaporedno številko $(2n+1)$ itd.

- [40%] `public static <T> Iterator<T> razmnozevalnik(List<T> seznam, int n)`

  Vrne iterator, ki na svoji poti enkrat obišče prvi element podanega seznama, dvakrat drugi, ..., $n$-krat $n$-ti element, enkrat element z zaporedno številko $(n+1)$, dvakrat element z zaporedno številko $(n+2)$ itd.

  Iterator se mora sprehajati po seznamu `seznam`. Metoda ne sme izdelati »razmnožene« kopije seznama in vrniti iteratorja, ki se trivialno sprehodi po kopiji!