# Drugi izpitni rok pri predmetu Programiranje 1
# 30. januar 2023

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`

*Število* je celo število, *beseda* pa neprazno zaporedje črk angleške abecede.

(1) V prvi vrstici vhoda je zapisano število $n \in [1, 10^5]$, nato pa sledi $n$ vrstic s pari števil z intervala $[1, 10^9]$. Napišite program (`Prva.java`), ki izpiše število vhodnih parov, v katerih je vsaka števka prvega števila strogo manjša od istoležne števke drugega števila. V vsakem paru imata obe števili enako število števk. V 30% testnih primerov so vsi pari sestavljeni iz dveh enomestnih števil.

Na primer, opisana lastnost velja za par (30742, 51869), saj je $3 < 5$, $0 < 1$, $7 < 8$, $4 < 6$ in $2 < 9$.

**Primer (vhod/izhod):**

```
5
1234 4321
7 9
30421 51846
467 568
467 578
```

```
3
```

(2) Elementi `true` v pravokotni tabeli `matrika` višine $h \in [3, 1000]$ in širine $w \in [3, 1000]$ tvorijo lomljenko, sestavljeno iz vodoravnih in navpičnih odsekov. Lomljenka se prične v celici v drugi vrstici in drugem stolpcu, zaključi pa v celici v predzadnji vrstici in predzadnjem stolpcu (ta dva elementa sta torej vedno enaka `true`). Vsi robni elementi tabele so enaki `false`.

V razredu `Druga` napišite metodo

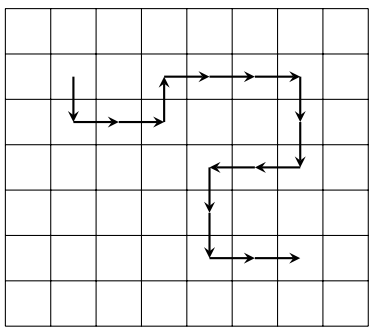
`public static int[] poLomljenki(boolean[][] matrika)`,

ki vrne tabelo $\{n_\leftarrow, n_\uparrow, n_\rightarrow, n_\downarrow\}$, kjer $n_\leftarrow$, $n_\uparrow$, $n_\rightarrow$ in $n_\downarrow$ podajajo število premikov v smeri levo, navzgor, desno oz. navzdol, ki jih opravimo, ko potujemo po lomljenki od njenega začetka do njenega konca. Lomljenka se nikjer ne seka, dotika pa se lahko kvečjemu diagonalno.

V 30% testnih primerov velja $n_\leftarrow = n_\uparrow = 0$.

V sledečem primeru (`T`: `true`; (prazno): `false`) bi metoda vrnila tabelo $\{2, 1, 7, 5\}$:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | T | | T | T | T | T | |
| | T | T | T | | | T | |
| | | | | T | T | T | |
| | | | | T | | | |
| | | | | T | T | T | |
| | | | | | | | |



Namig: vsebino tabele `matrika` lahko po potrebi spreminjate.

(3) V razredu `Tretja` so definirani sledeči statični notranji razredi:

```
class Igrisce {     // otroško igrišče
    private Igralo[] igrala;    // igrala na igrišču this
}
abstract class Igralo {
    private int x, y;    // položaj igrala this[1]

    // Vrne kvadrat minimalne dopustne evklidske razdalje med igralom this
    // in katerimkoli drugim igralom.
    public abstract int minRazdalja2();
}
class Gugalnica extends Igralo {}
class Tobogan extends Igralo {}
class Plezalo extends Igralo {}
class Vzmetnik extends Igralo {}  // npr. konjiček na vzmet
```

Dopolnite sledeče metode:

- `public int razdalja2(Igralo igr)` v razredu `Igralo`:

  Vrne kvadrat evklidske razdalje med igraloma `this` in `igr`.

- `public boolean poPredpisih()` v razredu `Igrisce`:

  Vrne `true` natanko v primeru, če je vsak par igral na igrišču `this` postavljen na zadostni medsebojni razdalji. Seveda morate pri vsakem paru igral upoštevati večjo izmed minimalnih dopustnih razdalj: če, denimo, minimalna dopustna razdalja za tobogan znaša 150, za vzmetnik pa 100, morata biti tobogan in vzmetnik med seboj oddaljena najmanj 150.

- `public int najTip()` v razredu `Igrisce`:

  Vrne številko (0: gugalnica; 1: tobogan; 2: plezalo; 3: vzmetnik) najbolj zastopanega tipa igral na igrišču `this`. V vsakem testnem primeru je najbolj zastopan natanko en tip.

(4) Slovar tipa `Map<String, Set<String>>` hrani medsebojno disjunktne množice sopomenk: vsaka beseda iz množice se preslika v to isto množico. Na primer, množici {veselje, radost} in {zopet, spet, znova} predstavimo s slovarjem {veselje ↦ {veselje, radost}, radost ↦ {veselje, radost}, zopet ↦ {zopet, spet, znova}, spet ↦ {zopet, spet, znova}, znova ↦ {zopet, spet, znova}}.

V razredu `Cetrta` dopolnite sledeči metodi:

- `public static Set<String> najMnozica(Map<String, Set<String>> sopomenke)`

  Vrne največjo množico sopomenk. V vsakem testnem primeru obstaja natanko ena taka množica.

- `public static Map<Set<String>, Integer> pogostost(`
  `String besedilo, Map<String, Set<String>> sopomenke)`

  Vrne slovar, ki vsako množico sopomenk preslika v njeno pogostost v podanem besedilu. Na primer, če se v vhodnem besedilu trikrat pojavi beseda `veselje`, štirikrat pa `radost`, je pogostost množice {veselje, radost} enaka 7.

  Besedilo je sestavljeno iz besed dolžine do 20, ki so med seboj ločene s po enim presledkom (namig: `split`). Besede, ki jih ni v vhodnem slovarju, ignorirajte.

[1] Za potrebe te naloge bomo predpostavili, da so igrala točke. Oprosti, realnost, a fiziki niso nič boljši!