Dodatne naloge pri predmetu Programiranje I

Predgovor

Pred vami je komplet nalog za utrjevanje snovi pri predmetu Programiranje 1. Dodatnih nalog ne bomo pregledovali in ocenjevali, saj so namenjene izključno utrjevanju snovi. Programiranje je namreč obrt, ki se je lahko priučite samo z rednim in sprotnim pisanjem programov. Zato vam priporočamo, da se takoj po predavanjih lotite nalog na temo odpredavane snovi. Naloge, označene z zvezdico, so nekoliko težje, a se jih nikar ne ustrašite. Vse je mogoče rešiti s programskimi konstrukti in tehnikami, ki jih bomo spoznali pri predmetu Programiranje 1.

Poleg besedila nalog boste na spletni učilnici našli tudi paket, ki vsebuje testne primere in morebitne izhodiščne datoteke za posamezne naloge. Vsaka dodatna naloga je opremljena s testnimi primeri, s katerimi lahko vaše rešitve preizkusite. Če smo natančnejši:

- Pri nalogah tipa vhod-izhod (vse naloge v poglavjih 1, 2, 3 ter naloge 4.10, 5.1, 5.2, 6.4 in 6.6) boste napisali samostojni program in ga testirali s pomočjo parov vhodnih datotek (`test*.in`) in pripadajočih izhodnih datotek (`test*.out`). Testni primer i se šteje kot pravilno obravnavan natanko tedaj, ko vaš program pri branju vhoda iz datoteke `test`i`.in` proizvede izpis, ki je enak vsebini datoteke `test`i`.out`.
- Pri nalogah tipa razred-izhod (naloge 4.1–4.9, 5.3, 5.4, 6.1–6.3, 6.5, 6.7, 6.8 in vse naloge v poglavju 7) boste napisali razred in ga testirali s pomočjo parov testnih razredov (datotek `Test*.java`) in pripadajočih izhodnih datotek (`test*.out`). Vsak testni razred praviloma ustvari vsaj en objekt razreda, ki ga preizkušate, nato pa na objektu kliče vsaj eno metodo in izpiše njen rezultat. Testni primer i se šteje kot pravilno obravnavan natanko tedaj, ko po zagonu razreda `Test`i`.java` dobimo izpis, ki je enak vsebini datoteke `test`i`.out`.

Pri nalogah, ki zahtevajo dopolnitev že delno napisanega razreda, boste v paketu na spletni učilnici našli tudi ustrezne izhodiščne razrede.

Veliko užitkov pri reševanju nalog!

Kazalo

1 Krmilni konstrukti

1.1 Absolutna vrednost

Naloga

Napišite program, ki prebere celo število in izpiše njegovo absolutno vrednost.

Vhod

Na vhodu je podano celo število z intervala $[-10^9, 10^9]$.

Izhod

Izpišite absolutno vrednost vhodnega števila. Izpis zaključite s skokom v naslednjo vrstico.[1]

Testni primer 1

Vhod:

```
-42
```

Izhod:

```
42
```

1.2 Trihotomija

Naloga

Napišite program, ki prebere dve celi števili in izpiše 1, če je prvo število večje od drugega, 0, če sta števili enaki, oziroma -1, če je prvo število majše od drugega.

Vhod

Na vhodu sta podani celi števili z intervala $[-10^9, 10^9]$, ločeni s presledkom.

[1] Pravila, da mora biti zadnji znak izhoda skok v naslednjo vrstico, se dosledno držimo pri vseh nalogah pri predmetu Programiranje 1. Absolutno vrednost boste torej izpisali z ukazom `System.out.println`, ne `System.out.print`.

Izhod

Izpišite 1, 0 oziroma -1.

Testni primer 1

Vhod:

```
7 10
```

Izhod:

```
-1
```

1.3 Časovna razlika I

Naloga

Napišite program, ki prebere pozitivna cela števila h_1, m_1, h_2 in m_2 in izpiše razliko (v minutah) med časoma h_1:m_1 in h_2:m_2. Na primer, razlika med časoma 15:58 in 18:04 ($h_1 = 15$, $m_1 = 58$, $h_2 = 18$, $m_2 = 4$) znaša 126 minut.

Vhod

Na vhodu so podana cela števila $h_1 \in [0, 23]$, $m_1 \in [0, 59]$, $h_2 \in [h_1, 23]$ in $m_2 \in [0, 59]$, ločena s presledkom. V primeru $h_1 = h_2$ velja $m_1 \leqslant m_2$.

Izhod

Izpišite iskano časovno razliko.

Testni primer 1

Vhod:

```
15 58 18 4
```

Izhod:

```
126
```

1.4 Časovna razlika II

Naloga

Napišite program, ki prebere pozitivna cela števila h_1, m_1, h_2 in m_2 in izpiše razliko (v urah in minutah) med časoma h_1:m_1 in h_2:m_2. Na primer, razlika med časoma 15:58 in 18:04 ($h_1 = 15$, $m_1 = 58$, $h_2 = 18$, $m_2 = 4$) znaša 2 uri in 6 minut.

Vhod

Na vhodu so podana cela števila $h_1 \in [0, 23]$, $m_1 \in [0, 59]$, $h_2 \in [h_1, 23]$ in $m_2 \in [0, 59]$, ločena s presledkom. V primeru $h_1 = h_2$ velja $m_1 \leqslant m_2$.

Izhod

Iskano časovno razliko izpišite v obliki `H:MM`, npr. `2:06` za 2 uri in 6 minut.

Testni primer 1

Vhod:

```
15 58 18 4
```

Izhod:

```
2:06
```

1.5 Najbližji večkratnik

Naloga

Napišite program, ki prebere števili a in b in izpiše tisti večkratnik števila a, ki je najmanj oddaljen od števila b. Če obstajata dva takšna večkratnika, naj izpiše manjšega od njiju.

Vhod

Na vhodu sta podani celi števili $a \in [1, 10^9]$ in $b \in [a, 10^9]$, ločeni s presledkom.

Izhod

Izpišite iskani večkratnik.

Testni primer 1

Vhod:

```
6 28
```

Izhod:

```
30
```

Testni primer 2

Vhod:

```
6 27
```

Izhod:

```
24
```

V tem primeru sta večkratnika 24 in 30 enako oddaljena od števila 27, vendar pa je prvi manjši.

1.6 Mediana trojice I

Naloga

Napišite program, ki prebere tri števila in izpiše srednje med njimi (tj. število, od katerega je vsaj eno od preostalih dveh števil v trojici manjše ali enako in vsaj eno večje ali enako).

Vhod

Na vhodu so podana tri cela števila z intervala $[-10^9, 10^9]$, ločena s presledkom.

Izhod

Izpišite iskano število.

Testni primer 1

Vhod:

```
7 10 8
```

Izhod:

```
8
```

Testni primer 2

Vhod:

```
5 3 5
```

Izhod:

```
5
```

1.7 Urejanje trojice

Naloga

Napišite program, ki prebere tri števila in jih izpiše v naraščajočem vrstnem redu.

Vhod

Na vhodu so podana tri cela števila z intervala $[-10^9, 10^9]$, ločena s presledkom.

Izhod

Izpišite vhodna števila v naraščajočem vrstnem redu. Med seboj jih ločite s presledkom.

Testni primer 1

Vhod:

```
5 7 2
```

Izhod:

```
2 5 7
```

Testni primer 2

Vhod:

```
-6 -9 -6
```

Izhod:

```
-9 -6 -6
```

1.8 Zaporedje zvezdic

Naloga

Napišite program, ki prebere število n in izpiše zaporedje n zvezdic.

Vhod

Na vhodu je podano celo število $n \in [1, 100]$.

Izhod

Izpišite zaporedje n zvezdic (znakov $*$). Izpis seveda tudi tokrat zaključite s skokom v naslednjo vrstico.

Testni primer 1

Vhod:

```
5
```

Izhod:

```
*****
```

1.9 Poštevanka I

Naloga

Napišite program, ki prebere števili a in b in po vrsti izpiše rezultate množenja števila a s števili od 1 do vključno b.

Vhod

Na vhodu sta zapisani celi števili $a \in [-10^6, 10^6]$ in $b \in [1, 10^3]$, ločeni s presledkom.

Izhod

Vsak rezultat množenja izpišite v svoji vrstici.

Testni primer 1

Vhod:

```
5 6
```

Izhod:

```
5
10
15
20
25
30
```

Testni primer 2

Vhod:

```
-20 3
```

Izhod:

```
-20
-40
-60
```

1.10 Poštevanka II

Naloga

Napišite program, ki prebere števili a in b in izpiše poštevanko števila a s faktorji od 1 do vključno b.

Vhod

Na vhodu sta zapisani celi števili $a \in [-10^6, 10^6]$ in $b \in [1, 10^3]$, ločeni s presledkom.

Izhod

Vsako enačbo poštevanke izpišite v svoji vrstici. Posamezno enačbo izpišite v sledeči obliki (znak $\sqcup$ predstavlja presledek):

$p \sqcup * \sqcup q \sqcup = \sqcup r$

Testni primer 1

Vhod:

```
5 6
```

Izhod:

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
```

Testni primer 2

Vhod:

```
-20 3
```

Izhod:

```
-20 * 1 = -20
-20 * 2 = -40
-20 * 3 = -60
```

1.11 Poštevanka III

Naloga

Napišite program, ki prebere števili a in b in izpisuje poštevanko števila a tako dolgo, dokler rezultat ni večji od b.

Vhod

Na vhodu sta zapisani celi števili $a \in [1, 10^9]$ in $b \in [a, 10^9]$, ločeni s presledkom.

Izhod

Vsako enačbo poštevanke izpišite v svoji vrstici, in sicer v sledeči obliki (znak $\sqcup$ predstavlja presledek):

$p\sqcup*\sqcup q\sqcup=\sqcup r$

Testni primer 1

Vhod:

```
6 25
```

Izhod:

```
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
```

Testni primer 2

Vhod:

```
5 25
```

Izhod:

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
```

1.12 Številska zaporedja

Naloga

Napišite program, ki prebere števila a, b in k in izpiše zaporedje števil od a do b s korakom k. V primeru $a \leqslant b$ naj se izpis zaključi pri največjem številu, ki ni večje od b, v primeru $a > b$ pa pri najmanjšem številu, ki ni manjše od b.

Pred izpisom zaporedja naj program preveri, ali vhod zadošča sledečima pogojema:

- korak k ni enak 0;

- korak je pozitiven v primeru $a < b$ oziroma negativen v primeru $a > b$.

Če vhod katerega od pogojev ne izpolnjuje, naj program izpiše zgolj besedo `NAPAKA`.

Vhod

Na vhodu so podana cela števila a, b in k, ločena s presledkom. Vsa tri števila pripadajo intervalu $[-10^9, 10^9]$.

Izhod

V primeru nepravilnega vhoda izpišite besedo `NAPAKA`, v primeru pravilnega vhoda pa ustrezno zaporedje. Vsako število izpišite v svojo vrstico.

Testni primer 1

Vhod:

```
10 30 0
```

Izhod:

```
NAPAKA
```

Testni primer 2

Vhod:

```
10 30 -2
```

Izhod:

```
NAPAKA
```

Testni primer 3

Vhod:

```
10 30 3
```

Izhod:

```
10
13
16
19
22
25
28
```

Testni primer 4

Vhod:

```
30 -20 -5
```

Izhod:

```
30
25
20
15
10
5
0
-5
-10
-15
-20
```

1.13 Potenca

Naloga

Napišite program, ki prebere števili a in b in izpiše vrednost potence a^b. Nalogo rešite s pomočjo zaporednih množenj, ne z metodo `Math.pow`.

Vhod

Na vhodu sta zapisani celi števili $a \in [1,\ 10^9]$ in $b \in [0,\ 100]$, ločeni s presledkom. Velja $a^b \leqslant 10^9$.

Izhod

Izpišite vrednost potence a^b.

Testni primer 1

Vhod:

```
3 4
```

Izhod:

```
81
```

1.14 EvroŠop®

Naloga

V trgovini EvroŠop® so vsi izdelki naprodaj za 1 evro. Vsaka stranka kupi samo po en izdelek, plača pa ga bodisi s kovancem za 1 evro ali pa s kovancem za 2 evra. V prvem primeru blagajničarka stranki seveda ne vrne ničesar (saj izdelek stane 1 evro), v drugem pa ji vrne kovanec za 1 evro. Blagajna je na začetku prazna.

Napišite program, ki prebere zaporedje podatkov o tem, s katerim kovancem je posamezna stranka plačala izdelek, nato pa izpiše končno število kovancev v blagajni. Lahko se zgodi, da blagajničarka stranki, ki je izdelek plačala s kovancem za 2 evra, ne more vrniti kovanca za 1 evro, ker jih v blagajni preprosto ni. V tem primeru naj se program zaključi z izpisom `BANKROT`.

Vhod

Na vhodu je zapisano zaporedje števil 1 in 2, ločenih s presledkom. Dolžina zaporedja ni znana vnaprej.

Izhod

Če blagajničarka neki stranki ne more vrniti denarja, izpišite samo besedo `BANKROT`. V nasprotnem primeru pa izpišite končno število kovancev v blagajni: v prvi vrstici izpišite število kovancev za 1 evro, v drugi pa za 2 evra.

Testni primer 1

Vhod:

```
1 1 1 1 1 2 1 1 1
```

Izhod:

```
7
1
```

V tem primeru najprej prejmemo 5 kovancev za 1 evro, nato pa enega vrnemo stranki, ki nam je dala kovanec za 2 evra. Nato prejmemo še 3 kovance za 1 evro.

Testni primer 2

Vhod:

```
1 2 2 1 1 2 2 2
```

Izhod:

```
BANKROT
```

V tem primeru že pri tretji stranki bankrotiramo.

1.15 Smučanje

Naloga

Na smučarskem tekmovanju nastopa n tekmovalcev. Tekmovalec odsmuča progo dvakrat, njegov rezultat pa je seštevek obeh časov. Če ga diskvalificirajo, se njegov rezultat ne upošteva. Če ga diskvalificirajo že v prvem teku, potem v drugem sploh ne bo nastopal.

Vaš program naj najprej prebere število n, nato pa za vsakega tekmovalca še njegov rezultat v prvem in drugem teku (če v prvem ni bil diskvalificiran). Rezultat je podan bodisi kot pozitivno celo število, ki podaja čas vožnje, ali pa kot število 0, ki pomeni diskvalifikacijo. Če so vse tekmovalce diskvalificirali, naj program to sporoči, sicer pa naj izpiše zaporedno številko tekmovalca z najboljšim skupnim časom in njegov skupni čas. Če je najboljših tekmovalcev več, naj program izbere tistega z najmanjšo zaporedno številko.

Vhod

V prvi vrstici vhoda je podano celo število $n \in [1, 10^6]$, nato pa sledi še n vrstic vhoda. V vsaki od teh n vrstic je podano bodisi samo število 0 ali pa dvoje celih števil, ločenih s presledkom. Prvo od teh dveh števil pripada intervalu $[1, 10^9]$, drugo pa intervalu $[0, 10^9]$.

Izhod

Če so vse tekmovalce diskvalificirali, izpišite samo niz `NIHCE`, v nasprotnem primeru pa v prvi vrstici izpišite zaporedno številko tekmovalca z najmanjšim skupnim časom, v drugi pa njegov skupni čas.

Testni primer 1

Vhod:

```
5
70 65
40 0
55 59
0
50 72
```

Izhod:

```
3
114
```

Tekmovalca 2 in 4 so diskvalificirali, zato ju ne upoštevamo, od preostalih pa je najboljši tekmovalec 3.

Testni primer 2

Vhod:

```
3
0
70 0
0
```

Izhod:

```
NIHCE
```

1.16 Vozni red

Naloga

Avtobus vozi v enakomernih časovnih presledkih. Napišite program, ki prebere čas začetka dnevne vožnje (h_z (ura) in m_z (minuta)), čas konca dnevne vožnje (h_k in m_k) in interval v minutah (d), nato pa izpiše dnevni vozni red. Prva vožnja se izvrši natanko ob času začetka vožnje, zadnja pa ob času, ki je kvečjemu enak času konca vožnje.

Vhod

Na vhodu je zapisanih pet celih števil, ločenih s presledkom: $h_z \in [0, 23]$, $m_z \in [0, 59]$, $h_k \in [h_z, 23]$, $m_k \in [0, 59]$ in $d \in [1, 1440]$. V primeru $h_z = h_k$ velja $m_z \leqslant m_k$.

Izhod

Izpišite odhode avtobusa v naraščajočem vrstnem redu. Vsak odhod izpišite v svoji vrstici. Posamezni odhodi naj bodo zapisani v obliki `HH:MM` (npr. `09:05` za pet čez deveto).

Testni primer 1

Vhod:

```
10 0 15 0 30
```

Izhod:

```
10:00
10:30
11:00
11:30
12:00
12:30
13:00
13:30
14:00
14:30
15:00
```

Testni primer 2

Vhod:

```
8 50 14 10 35
```

Izhod:

```
08:50
09:25
10:00
10:35
11:10
11:45
12:20
12:55
13:30
14:05
```

1.17 Piramida števil

Naloga

Napišite program, ki prebere število n in nariše »piramido« števil višine n, kot jo prikazujeta primera v nadaljevanju.

Vhod

Na vhodu je podano celo število $n \in [1, 100]$.

Izhod

Izpišite »piramido« po zgledu sledečih primerov. Ne izpisujte odvečnih presledkov in praznih vrstic.

Testni primer 1

Vhod:

```
5
```

Izhod:

```
    1
   234
  34567
 4567890
567890123
```

Izhod s prikazanimi presledki:

```
␣␣␣␣1
␣␣␣234
␣␣34567
␣4567890
567890123
```

Testni primer 2

Vhod:

```
11
```

Izhod:

```
          1
         234
        34567
       4567890
      567890123
     67890123456
    7890123456789
   890123456789012
  90123456789012345
 0123456789012345678
12345678901234567901
```

1.18 Igorjevi bloki

Naloga

Napišite program, ki prebere tri enomestna števila in nariše vzorec, kot ga prikazujeta primera v nadaljevanju.

Vhod

Na vhodu so podana tri cela števila z intervala $[1, 9]$, ločena s presledkom.

Izhod

Izpišite vzorec po zgledu sledečih primerov. Ne izpisujte odvečnih presledkov in praznih vrstic.

Testni primer 1

Vhod:

```
3 7 4
```

Izhod:

```
333 7777777 4444
333 7777777 4444
333 7777777 4444
    7777777 4444
    7777777
    7777777
    7777777
```

Izhod s prikazanimi presledki:

```
333␣7777777␣4444
333␣7777777␣4444
333␣7777777␣4444
␣␣␣␣7777777␣4444
␣␣␣␣7777777
␣␣␣␣7777777
␣␣␣␣7777777
```

Testni primer 2

Vhod:

```
2 2 3
```

Izhod:

```
22 22 333
22 22 333
      333
```

1.19 Šahovnica

Naloga

Napišite program, ki prebere števila v, s in d in nariše vzorec v obliki šahovnice z v vrsticami in s stolpci, pri čemer ima vsako polje obliko kvadrata velikosti $d \times d$. Šahovnica naj bo tudi obrobljena. Zgledujte se po primerih v nadaljevanju.

Vhod

Na vhodu so podana cela števila v, s in d z intervala $[1, 20]$. Med seboj so ločena s presledkom.

Izhod

Izpišite vzorec po zgledu sledečih primerov. Ne izpisujte odvečnih presledkov in praznih vrstic.

Testni primer 1

Vhod:

```
3 4 5
```

Izhod:

```
+ - - - - - - - - - - - - - - - - - - - - +
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
| * * * * *           * * * * *           |
| * * * * *           * * * * *           |
| * * * * *           * * * * *           |
| * * * * *           * * * * *           |
| * * * * *           * * * * *           |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
|           * * * * *           * * * * * |
+ - - - - - - - - - - - - - - - - - - - - +
```

Testni primer 2

Vhod:

```
6 5 2
```

Izhod:

```
+ - - - - - - - - - - +
|     * *     * *     |
|     * *     * *     |
| * *     * *     * * |
| * *     * *     * * |
|     * *     * *     |
|     * *     * *     |
| * *     * *     * * |
| * *     * *     * * |
|     * *     * *     |
|     * *     * *     |
| * *     * *     * * |
| * *     * *     * * |
+ - - - - - - - - - - +
```

Izhod s prikazanimi presledki:

```
+␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣+
|␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣*␣*␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
```

```
|␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
|␣␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
|␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
|␣␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣|
|␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
|␣*␣*␣␣␣␣␣*␣*␣␣␣␣␣*␣*␣|
+␣-␣-␣-␣-␣-␣-␣-␣-␣-␣-␣+
```

1.20 Anžetove ledene sveče

Naloga

Napišite program, ki prebere število n in nariše vzorec višine n, kot ga prikazujejo primeri v nadaljevanju.

Vhod

Na vhodu je podano celo število $n \in [2, 20]$.

Izhod

Izpišite vzorec po zgledu sledečih primerov. Ne izpisujte odvečnih presledkov in praznih vrstic.

Testni primer 1

Vhod:

```
5
```

Izhod:

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

Testni primer 2

Vhod:

```
6
```

Izhod:

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

Testni primer 3

Vhod:

```
7
```

Izhod:

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

1.21 Metaprogram

Naloga

S sledečo zanko po vrsti izpišemo vse velike črke angleške abecede od A do Z:

```
for (char c1 = 'A';  c1 <= 'Z';  c1++) {
    System.out.println("" + c1);
}
```

Sedaj pa bi želeli po abecednem vrstnem redu izpisati vse nize (besede), sestavljene iz n velikih črk angleške abecede. Na primer, če je $n = 2$, bi želeli izpisati nize `AA`, `AB`, ..., `AZ`, `BA`, `BB`, ..., `BZ`, ..., `ZA`, `ZB`, ..., `ZZ`. Če je $n = 3$, bi želeli izpisati nize `AAA`, `AAB`, ..., `ZZZ`. Da bi lahko nalogo rešili za poljuben n, bi potrebovali nekoliko več znanja, kot ga imamo sedaj. Za fiksen n pa lahko nalogo rešimo s pomočjo n vgnezdenih zank. Ker program seveda ne more vsebovati spremenljivega števila zank, boste napisali *metaprogram* — program, ki izpiše program, ki opisani problem reši s pomočjo n vgnezdenih zank. Natančno se zgledujte po primeru, prikazanem v nadaljevanju.

Vhod

Na vhodu je podano celo število $n \in [1, 100]$.

Izhod

Izpišite program po zgledu sledečega testnega primera. Natančno se držite števila presledkov med posameznimi elementi izpisanega programa. Ne izpisujte tabulatorjev, odvečnih presledkov in praznih vrstic.

Testni primer 1

Vhod:

```
4
```

Izhod:

```
public class Nizi {
    public static void main(String[] args) {
        for (char c1 = 'A';  c1 <= 'Z';  c1++) {
```

```
            for (char c2 = 'A';  c2 <= 'Z';  c2++) {
                for (char c3 = 'A';  c3 <= 'Z';  c3++) {
                    for (char c4 = 'A';  c4 <= 'Z';  c4++) {
                        System.out.println("" + c1 + c2 + c3 + c4);
                    }
                }
            }
        }
    }
}
```

Izhod s prikazanimi presledki:

```
public␣class␣Nizi␣{
␣␣␣␣public␣static␣void␣main(String[]␣args)␣{
␣␣␣␣␣␣␣␣for␣(char␣c1␣=␣'A';␣␣c1␣<=␣'Z';␣␣c1++)␣{
␣␣␣␣␣␣␣␣␣␣␣␣for␣(char␣c2␣=␣'A';␣␣c2␣<=␣'Z';␣␣c2++)␣{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣(char␣c3␣=␣'A';␣␣c3␣<=␣'Z';␣␣c3++)␣{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣for␣(char␣c4␣=␣'A';␣␣c4␣<=␣'Z';␣␣c4++)␣{
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣System.out.println(""␣+␣c1␣+␣c2␣+␣c3␣+␣c4);
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣}
␣␣␣␣␣␣␣␣␣␣␣␣}
␣␣␣␣␣␣␣␣}
␣␣␣␣}
}
```

Napotek

Če želite *izpisati* enojne ali dvojne navednice s pomočjo ukazov `System.out.print*`, uporabite zaporedje `\"` oz. `\'`. Na primer, ukaz

```
System.out.println("Znak \'a\' nastopa v nizu \"miza\".");
```

izpiše besedilo

```
Znak 'a' nastopa v nizu "miza".
```

1.22 Razbijanje števil

Naloga

Napišite program, ki prebere števili n in m in po vrsti izpiše posamezne dele (zaporedja števk) števila n, pri čemer je dolžina posameznega dela določena s pripadajočo števko v številu m. Dolžina prvega dela je tako enaka prvi števki števila m, dolžina drugega dela je enaka drugi števki števila m itd.

Nalogo rešite zgolj z operacijami nad celimi števili. Uporaba realnoštevilskih operacij ter nizov, tabel ipd. ni dovoljena.

Vhod

Na vhodu sta podani celi števili $n \in [1, 10^{18}]$ in $m \in [1, 10^{18}]$, ločeni s presledkom. Vsota števk števila m je enaka številu števk števila n. Števili n in m ne vsebujeta nobene ničle.

Izhod

Izpišite toliko vrstic, kolikor je števk števila m. V prvi vrstici izpišite začetnih a_1 števk števila n (pri čemer je a_1 prva števka števila m), v drugi sledečih a_2 števk števila n (pri čemer je a_2 druga števka števila m) itd.

Testni primer 1

Vhod:

```
3629831574865 2317
```

Izhod:

```
36
298
3
1574865
```

2 Metode

2.1 Predvolilni golaž

Naloga

Politiku Gvidu[1] je podpora pred volitvami nevarno padla, zato se odloči, da bo izbranim skupinam volilcev plačeval kosila v dobrih gostilnah tako dolgo, dokler mu ne zmanjka denarja. Vsak dan povabi določeno skupino ljudi v izbrano gostilno. Cena pogostitve se v osnovi izračuna kot zmnožek števila kosil in cene kosila, izbrano vino pa ceno poveča za navzgor zaokroženo polovico (če cena celotne pogostitve brez vina znaša 45 evrov, je cena z vinom enaka $45 + 23 = 68$ evrov, če pa bi brez vina odšteli 46 evrov, bi z vinom $46 + 23 = 69$ evrov). Napišite program, ki najprej prebere podatek o začetni zalogi Gvidovega denarja, nato pa zaporedoma bere podatke o pogostitvah ter sproti izpisuje njihove cene in preostalo zalogo denarja. Program naj se zaključi, ko zmanjka vhoda ali pa Gvidovega denarja.

V programu definirajte in uporabite metodo, ki sprejme podatke o pogostitvi in vrne njeno ceno.

Vhod

V prvi vrstici vhoda je zapisana začetna količina Gvidovega denarja (celo število z intervala $[0, 10^9]$), v vseh ostalih vrsticah pa so zapisani podatki o posameznih pogostitvah. Število pogostitev ni znano vnaprej. Vsaka pogostitev je opredeljena s tremi števili, ki so med seboj ločena s presledkom:

- cena enega kosila (celo število z intervala $[0, 10^3]$);
- število kosil (celo število z intervala $[1, 10^3]$);
- podatek o tem, ali so gostje pili tudi izbrano vino (1: da, 0: ne).

Izhod

V i-ti vrstici izhoda izpišite dve števili, ločeni s presledkom: ceno i-te pogostitve in zalogo Gvidovega denarja po i-ti pogostitvi. Če Gvidu zmanjka denarja za plačilo trenutne pogostitve, namesto drugega števila izpišite znak - (minus).

[1] Po istoimenski skladbi Iztoka Mlakarja

Testni primer 1

Vhod:

```
500
30 5 0
30 6 1
15 5 1
20 7 0
50 4 1
```

Izhod:

```
150 350
270 80
113 -
```

V tem primeru Gvido ostane brez denarja še pred koncem vhoda.

Testni primer 2

Vhod:

```
1000
30 5 0
30 6 1
15 5 1
20 7 0
50 4 1
```

Izhod:

```
150 850
270 580
113 467
140 327
300 27
```

V tem primeru Gvido »zdrži« do konca vhoda.

2.2 Množenje z zaporednim seštevanjem

Naloga

Napišite program, ki prebere dve števili in izpiše njun zmnožek. Definirajte in smiselno uporabite metodo, ki sprejme dve celi števili in vrne njuno vsoto.[2]

Vhod

Na vhodu sta podani celi števili z intervala $[1, 1000]$.

Izhod

Izpišite zmnožek vhodnih števil.

[2] Množenje z zaporednim seštevanjem je seveda neučinkovito, a pri tej nalogi se osredotočamo na pisanje in klicanje metod.

Testni primer 1

Vhod:

```
6 7
```

Izhod:

```
42
```

2.3 Mediana trojice II

Naloga

Napišite program, ki prebere tri števila in izpiše srednje med njimi (tj. število, od katerega je vsaj eno od preostalih dveh števil v trojici manjše ali enako in vsaj eno večje ali enako). Definirajte in smiselno uporabite metodi `min` in `maks`, ki sprejmeta dve celi števili in vrneta manjše (`min`) oziroma večje (`maks`) izmed njiju.[3]

Vhod

Na vhodu so podana tri cela števila z intervala $[-10^9, 10^9]$, ločena s presledkom.

Izhod

Izpišite iskano število.

Testni primer 1

Vhod:

```
7 10 8
```

Izhod:

```
8
```

Testni primer 2

Vhod:

```
5 3 5
```

Izhod:

```
5
```

2.4 Štetje klicev I (⋆)

Naloga

Podani sta celi števili $a \geqslant 2$ in $b \geqslant 2$. Funkcija $f\colon \mathbb{N}_0 \to \mathbb{N}_0$ je definirana takole:

$$f(n) = \begin{cases} 1 & \text{pri } n = 0 \\ f(\lfloor \frac{n}{a} \rfloor) + f(\lfloor \frac{n}{b} \rfloor) & \text{pri } n > 0 \end{cases}$$

[3] Java premore vgrajeni metodi `Math.min` in `Math.max`, a tokrat napišite svoji.

Zapis $\lfloor r \rfloor$ označuje celi del realnega števila r (npr. $\lfloor 2{,}8 \rfloor = 2$).

Napišite program, ki prebere števila a, b in n in izpiše število klicev funkcije f, če vrednost $f(n)$ izračunamo strogo po definiciji.

Vhod

Na vhodu so podana cela števila $a \in [2,\ 100]$, $b \in [2,\ 100]$ in $n \in [0,\ 10^6]$, ločena s presledkom.

Izhod

Izpišite število klicev funkcije. To število bo zanesljivo manjše od 10^9.

Testni primer 1

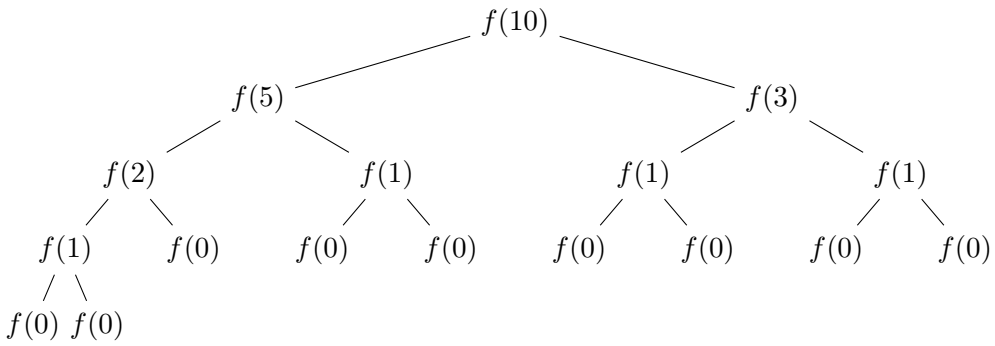
Vhod:

```
2 3 10
```

Izhod:

```
17
```

V tem primeru se funkcija f pokliče 17-krat:



2.5 Potenca po modulu (⋆)

Naloga

Napišite program, ki prebere števila a, b in m in izpiše rezultat izraza $a^b \bmod m$.

Vhod

Na vhodu so zapisana cela števila $a \in [1,\ 10^9]$, $b \in [0,\ 10^9]$ in $m \in [2,\ 10^9]$, ločena s presledkom.

Izhod

Izpišite samo rezultat.

Testni primer 1

Vhod:

```
5 2 7
```

Izhod:

```
4
```

Namiga

- Upoštevajte, da velja $pq \bmod m = (p \bmod m)(q \bmod m) \bmod m$.
- Kako izračunamo a^b, če poznamo $a^{\lfloor b/2 \rfloor}$?

2.6 Najboljše seme

Naloga

Pri tvorbi naključnih števil s pomočjo objekta razreda `java.util.Random` lahko podamo t.i. *seme* naključnega generatorja. Seme je število, ki enolično določa zaporedje naključnih števil, ki jih (na nekem fiksnem intervalu) tvori naključni generator. Na primer, sledeči program bo vedno izpisal zaporedje $\langle 0, 3, 8, 4, 0 \rangle$, ne glede na to, kolikokrat ga izvršimo:

```
import java.util.Random;

public class Program {

    public static void main(String[] args) {
        Random rand = new Random(42);
        for (int i = 1;  i <= 5;  i++) {
            System.out.println(rand.nextInt(10));
        }
    }
}
```

Seme podamo ob izdelavi objekta tipa `Random`, klic `rand.nextInt(`k`)` pa vrne naključno število med 0 in vključno $k - 1$.

Recimo, da tvorimo n naključnih števk med 0 in vključno 9 in jih sestavimo v število. V gornjem primeru ($seme = 42$, $n = 5$) bi na ta način dobili število 03840 oziroma 3840 (začetne ničle seveda nimajo učinka). Če bi vzeli seme 63, bi dobili zaporedje števk $\langle 9, 9, 9, 4, 9 \rangle$ in s tem število 99949. To število je največje med števili, ki jih po opisanem postopku dobimo s semeni med 1 in vključno 100.

Napišite program, ki prebere števila a, b in n in izpiše seme med a in vključno b, ki po opisanem postopku privede do največjega števila. Če je takih semen več, naj izpiše najmanjše med njimi.

Vhod

Na vhodu so podana cela števila $a \in [1, 10^5]$, $b \in [a, 10^5]$ in $n \in [1, 18]$, ločena s presledkom.

Izhod

Izpišite iskano seme.

Testni primer 1

Vhod:

```
1 100 5
```

Izhod:

```
63
```

2.7 Vraževerni Boris

Naloga

Boris vsak dan ponavlja sledeči obred za odganjanje zlih duhov: igralno kocko meče tako dolgo, dokler trikrat (v celotnem obredu) ne pade liho število pik. Ob nedeljah obred zaključi šele, ko liho število pik pade petkrat. Napišite program, ki prebere seme naključnega generatorja (s) in pozitivno celo število (n), nato pa simulira Borisov obred za n dni, začenši s ponedeljkom. Za vsak dan naj program izpiše še skupno število metov kocke.

Navodila za uporabo semena naključnega generatorja so podana v razdelku *Napotek* ob koncu te naloge.

Vhod

Na vhodu sta podani celi števili $s \in [1, 10^9]$ in $n \in [1, 10^3]$, ločeni s presledkom.

Izhod

Na izhod izpišite n vrstic. Vsaka vrstica naj bo zapisana v formatu

D␣`(`T`):`␣M␣`[`S`]`

pri čemer je

- D zaporedna številka dneva, zapisana na 4 mesta (npr. ␣␣␣`3` ali ␣`735`);
- T oznaka dneva v tednu (`D`: delovnik ali sobota, `N`: nedelja);
- M zaporedje rezultatov metov kocke, ločenih s presledkom;
- S skupno število metov v tekočem dnevu.

Testni primer 1

Vhod:

```
123456 22
```

Izhod:

```
   1 (D): 4 6 2 2 5 2 4 5 1 [9]
   2 (D): 2 4 3 4 1 1 [6]
   3 (D): 4 5 3 2 2 4 1 [7]
   4 (D): 4 3 1 4 5 [5]
```

```
  5 (D): 6 3 1 4 1 [5]
  6 (D): 6 3 4 5 5 [5]
  7 (N): 4 4 6 6 6 5 3 3 5 6 3 [11]
  8 (D): 3 6 2 3 4 1 [6]
  9 (D): 4 3 6 2 1 3 [6]
 10 (D): 6 1 5 1 [4]
 11 (D): 3 4 1 5 [4]
 12 (D): 5 1 2 4 1 [5]
 13 (D): 3 6 4 6 1 6 6 4 4 3 [10]
 14 (N): 3 6 6 4 3 3 6 4 4 3 4 4 1 [13]
 15 (D): 2 5 3 6 2 3 [6]
 16 (D): 4 4 6 1 1 2 4 6 5 [9]
 17 (D): 6 2 2 1 3 4 4 6 4 3 [10]
 18 (D): 6 5 1 4 4 2 2 3 [8]
 19 (D): 6 2 6 2 3 1 2 1 [8]
 20 (D): 1 2 3 5 [4]
 21 (N): 3 6 5 6 3 1 1 [7]
 22 (D): 4 4 5 3 3 [5]
```

Napotek

Naključne rezultate metov tvorite s pomočjo razreda `Random` iz paketa `java.util`. Natančno se držite sledeče sheme, sicer se vaši izhodi ne bodo ujemali s testnimi:

```
// na začetku
Random rand = new Random(s);

// pri tvorbi posameznih rezultatov metov
int stPik = rand.nextInt(a) + b;
```

Pri tem je s prebrano seme, a in b pa sta celi števili, ki jima morate seveda določiti ustrezni vrednosti.

2.8 Zdolgočasena Mojca

Naloga

Mojca preganja dolgčas z metanjem igralnih kock. V vsakem metu hkrati vrže k kock. Vsak dan izvaja mete tako dolgo, dokler ni vsota vseh k kock v tekočem metu praštevilo. Napišite program, ki s pomočjo generatorja naključnih števil simulira Mojčino igro za d dni.

Vhod

Na vhodu je najprej podano seme generatorja naključnih števil (celo število z intervala $[1, 10^9]$), nato pa še celi števili $k \in [1, 100]$ in $d \in [1, 100]$. Števila so ločena s presledkom.

Izhod

Za vsak dan simulacije naj program najprej izpiše vrstico oblike

D`.␣dan:`

kjer je D zaporedna številka tekočega dne, nato pa naj izpiše podatke o posameznih metih. Za vsak met naj se izpiše vrstica oblike

␣␣␣␣M`.␣met:␣`*kocke*`␣|␣vsota␣=␣`V

kjer je M zaporedna številka meta, *kocke* zaporedje s presledkom ločenih števil, ki podajajo število pik na posameznih kockah, V pa vsota pik za tekoči met.

Testni primer 1

Vhod:

```
12345 5 7
```

Izhod:

```
1. dan:
    1. met: 2 5 4 1 2 | vsota = 14
    2. met: 5 2 1 2 4 | vsota = 14
    3. met: 6 1 5 5 3 | vsota = 20
    4. met: 5 6 2 3 4 | vsota = 20
    5. met: 6 2 5 5 3 | vsota = 21
    6. met: 4 6 6 6 4 | vsota = 26
    7. met: 4 6 4 2 6 | vsota = 22
    8. met: 6 3 4 5 5 | vsota = 23
2. dan:
    1. met: 1 5 4 3 3 | vsota = 16
    2. met: 5 1 2 3 1 | vsota = 12
    3. met: 1 5 1 5 3 | vsota = 15
    4. met: 4 3 4 5 4 | vsota = 20
    5. met: 1 4 2 5 6 | vsota = 18
    6. met: 2 1 2 1 2 | vsota = 8
    7. met: 3 6 5 5 5 | vsota = 24
    8. met: 1 4 5 3 2 | vsota = 15
    9. met: 4 5 1 1 4 | vsota = 15
    10. met: 5 4 6 3 5 | vsota = 23
3. dan:
    1. met: 2 2 2 4 2 | vsota = 12
    2. met: 4 2 3 1 2 | vsota = 12
    3. met: 5 5 4 6 5 | vsota = 25
    4. met: 2 3 4 1 3 | vsota = 13
4. dan:
    1. met: 3 4 3 3 1 | vsota = 14
    2. met: 3 3 4 3 4 | vsota = 17
5. dan:
    1. met: 3 2 5 6 2 | vsota = 18
    2. met: 1 6 2 2 4 | vsota = 15
    3. met: 3 1 4 5 6 | vsota = 19
6. dan:
    1. met: 6 3 2 6 6 | vsota = 23
7. dan:
    1. met: 3 3 2 1 3 | vsota = 12
    2. met: 3 4 3 3 6 | vsota = 19
```

Napotek

Generator naključnih števil uporabite na enak način kot v nalogi Vraževerni Boris.

2.9 Šahovski popoldnevi

Naloga

Andrej in Branko vsak dan šahirata. Vsaka njuna partija se z verjetnostjo $a\%$ zaključi z zmago Andreja, z verjetnostjo $b\%$ z zmago Branka in s preostalo verjetnostjo z remijem. Njun dnevni dvoboj se zaključi po p partijah oziroma takrat, ko eden od njiju nabere z zmag (odvisno od tega, kaj se zgodi prej).

Napišite program, ki s pomočjo generatorja naključnih števil izvede simulacijo opisanega šahovskega dvoboja za d dni. Program naj za vsak dan izpiše zaporedje izidov in število odigranih partij.

Vhod

Na vhodu so zapisana cela števila $s \in [1, 10^9]$ (seme generatorja naključnih števil), $a \in [0, 100]$, $b \in [0, 100 - a]$, $p \in [1, 10^3]$, $z \in [0, 10^3]$ in $d \in [1, 100]$, ločena s presledkom.

Izhod

Za vsak dan izpišite po eno vrstico v obliki

D`␣dan:␣`*zaporedje*`␣(`N`)`

kjer je D številka tekočega dneva, *zaporedje* zaporedje znakov `A`, `B` in `r`, ki ponazarjajo izide posameznih partij v tekočem dnevu (`A`: zmaga Andreja; `B`: zmaga Branka; `r`: remi), N pa število odigranih partij v tekočem dnevu.

Testni primer 1

Vhod:

```
12345 50 30 8 3 10
```

Izhod:

```
1. dan: BrAABrB (7)
2. dan: AArA (4)
3. dan: ArAA (4)
4. dan: rrAABA (6)
5. dan: BrAArA (6)
6. dan: ABAA (4)
7. dan: rAAA (4)
8. dan: BBAAA (5)
9. dan: rrrBArAr (8)
10. dan: BABAB (5)
```

V 9. dnevu sta Andrej in Branko zaključila po skupno p (osmih) odigranih partijah, v vseh ostalih dneh pa po k (treh) zmagah enega od njiju.

Napotek

Verjetnostno pogojene izide generirajte tako, da tvorite naključno celo število med 0 in vključno 99 (s pomočjo metode `nextInt`, ki jo kličete na objektu razreda `Random`), nato pa upoštevate sledeča pravila:

- če je dobljeno število manjše od a, zmaga Andrej;
- če je dobljeno število v intervalu $[a,\ a+b)$, zmaga Branko;
- sicer pa se partija zaključi z remijem.

3 Tabele

3.1 Najbližji element

Naloga

Napišite program, ki prebere število k, število n in zaporedje n števil in izpiše indeks tistega elementa zaporedja, ki je od števila k najmanj oddaljen (ni pomembno, ali v pozitivno ali v negativno smer). Če je takih elementov več, naj program izpiše indeks prvega od njih.

Indeksi se pričnejo z ničlo; prvi element zaporedja ima tako indeks 0, drugi 1 itd.

Vhod

V prvi vrstici je podano celo število $k \in [-10^9, 10^9]$, v drugi celo število $n \in [1, 10^4]$, v tretji pa zaporedje n celih števil z intervala $[-10^9, 10^9]$, ločenih s presledkom.

Izhod

Izpišite samo iskani indeks (z metodo `System.out.println`, kot smo že navajeni).

Testni primer 1

Vhod:

```
45
6
30 90 60 40 -10 50
```

Izhod:

```
3
```

Od števila 45 sta najmanj oddaljena elementa na indeksih 3 (= 40) in 5 (= 50). Prvi med njima ima indeks 3.

3.2 Digitalne črtice

Naloga

Napišite program, ki prebere zaporedje pozitivnih celih števil in izpiše, katero od njih bi bilo na kalkulatorju s klasičnim digitalnim prikazovalnikom zapisano z največ črticami. Če

je takih števil v zaporedju več, naj izpiše prvo od njih.

Sledeča preglednica podaja število črtic, iz katerih so sestavljene posamezne števke:

Števka	0	1	2	3	4	5	6	7	8	9
Število črtic	6	2	5	5	4	5	6	3	7	6

Vhod

V prvi vrstici je zapisano celo število $n \in [1, 10^3]$, v drugi pa zaporedje n celih števil z intervala $[1, 10^9]$, ločenih s presledkom.

Izhod

Izpišite samo iskani element zaporedja.

Testni primer 1

Vhod:

```
8
4567 888 1113111 90 2352 211 9 63
```

Izhod:

```
888
```

Testni primer 2

Vhod:

```
3
35 61 127
```

Izhod:

```
35
```

Števili 35 in 127 sta obe zapisani z 10 črticami, program pa izpiše 35, ker se v zaporedju pojavi prej.

3.3 Pascalov trikotnik

Naloga

Napišite program, ki prebere celo število n in izpiše Pascalov trikotnik višine $n + 1$. V Pascalovem trikotniku je i-ta vrstica (za $i \in \{1, \ldots, n + 1\}$) sestavljena iz i števil. V vsaki vrstici sta prvo in zadnje število enaki 1, za $1 < j < i$ pa se število $p_{i,\,j}$ (j-to število v i-ti vrstici) izračuna po formuli $p_{i,\,j} = p_{i-1,\,j-1} + p_{i-1,\,j}$.

Vhod

Na vhodu je podano samo celo število $n \in [0, 30]$.

Izhod

Izpišite Pascalov trikotnik višine $n + 1$. Vsako vrstico trikotnika izpišite v svoji vrstici. Števila znotraj iste vrstice naj bodo ločena s presledkom. Na koncu vrstic ne sme biti presledkov!

Testni primer 1

Vhod:

```
4
```

Izhod:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Izhod s prikazanimi presledki:

```
1
1␣1
1␣2␣1
1␣3␣3␣1
1␣4␣6␣4␣1
```

Testni primer 2

Vhod:

```
10
```

Izhod:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1
```

3.4 Telefonski imenik

Naloga

Napišite program, ki najprej prebere n osebnih imen in pripadajočih telefonskih številk, nato pa prebere še k osebnih imen in za vsako izpiše pripadajočo telefonsko številko. Za imena, ki ne nastopajo med n imeni s pripisanimi telefonskimi številkami, naj program

izpiše niz `NEZNANA`. V seznamu imen s pripisanimi telefonskimi številkami se lahko imena tudi ponavljajo; v tem primeru velja zadnja telefonska številka.

Vhod

V prvi vrstici je podano celo število $n \in [0, 10^3]$. V vsaki od naslednjih n vrstic je najprej zapisano osebno ime v obliki niza do 20 črk angleške abecede, nato pa sledita presledek in pripadajoča telefonska številka v obliki niza do 20 števk ter znakov `+` in `-`. V naslednji vrstici je zapisano celo število $k \in [1, 10^3]$. Sledi še k vrstic, od katerih je v vsaki zapisano samo osebno ime v obliki niza do 20 črk angleške abecede.

Izhod

Izpišite k vrstic. V i-ti vrstici izpišite telefonsko številko, ki pripada imenu v i-ti vrstici znotraj skupine vrstic brez pripisanih telefonskih številk. Če ime ne nastopa v skupini vrstic s pripisanimi telefonskimi številkami, izpišite niz `NEZNANA`.

Testni primer 1

Vhod:

```
7
Mojca 01-234-567
Peter 041-317-650
Ivan +386-31-55-72-08
Helga +49-11-22-33-44-55
Polona 059-456-789
Helga +49-66-77-88-99-00
Ivan 031-78-56-34
6
Polona
Mirko
Helga
Iva
Ivan
Mojca
```

Izhod:

```
059-456-789
NEZNANA
+49-66-77-88-99-00
NEZNANA
031-78-56-34
01-234-567
```

Napotek

Nize berete podobno kot cela števila, le da namesto klica `sc.nextInt()` (kjer je `sc` objekt tipa `Scanner`) uporabite klic `sc.next()`. Za primerjanje nizov uporabite metodo `equals` (`niz1.equals(niz2)`), ne dvojnega enačaja.

3.5 Zlata sredina

Naloga

Napišite program, ki prebere število k in zaporedje $(2k + 1)$ medsebojno različnih celih števil in izpiše element zaporedja, od katerega je k elementov manjših in k večjih.

Vhod

V prvi vrstici je podano celo število $k \in [0, 10^5]$, v drugi pa zaporedje $(2k + 1)$ medsebojno različnih celih števil z intervala $[-10^9, 10^9]$, ločenih s presledkom.

Izhod

Izpišite samo iskani element zaporedja.

Testni primer 1

Vhod:

```
5
2 10 8 4 9 -6 6 1 3 -4 -2
```

Izhod:

```
3
```

3.6 Vsi različni I

Naloga

Napišite program, ki prebere zaporedje celih števil in izpiše `RAZLICNI`, če so vsi elementi v njem medsebojno različni. V nasprotnem primeru naj izpiše najmanjše število, ki v zaporedju nastopa najmanj dvakrat.

Vhod

V prvi vrstici je podano celo število $n \in [1, 10^4]$, v drugi pa zaporedje n celih števil z intervala $[-10^9, 10^9]$, ločenih s presledkom.

Izhod

Izpišite samo niz `RAZLICNI` oziroma najmanjše število, ki se v zaporedju ponovi.

Testni primer 1

Vhod:

```
6
3 9 8 10 6 2
```

Izhod:

```
RAZLICNI
```

Testni primer 2

Vhod:

```
7
3 10 9 8 9 2 10
```

Izhod:

```
9
```

3.7 Vsi različni II

Naloga

Napišite program, ki za zaporedje celih števil, tvorjeno z naključnim generatorjem, izpiše `RAZLICNI`, če so vsi elementi v njem medsebojno različni. V nasprotnem primeru naj izpiše število, ki v zaporedju največkrat nastopa. Če je takih števil več, naj izpiše najmanjše izmed njih.

Vhod

Na vhodu sta podani celi števili $s \in [1, 10^9]$ (seme generatorja naključnih števil) in $n \in [1, 10^7]$ (dolžina zaporedja). Naključni generator uporabite tako:

```
// na vrhu datoteke
import java.util.Random;

// takoj za glavo razreda
private static final int MAKS_STEVILO = 10000;

// pred pričetkom tvorbe zaporedja
Random random = new Random(seme);

// vsakokrat, ko tvorite člen zaporedja
int clen = random.nextInt(2 * MAKS_STEVILO + 1) - MAKS_STEVILO;
```

Vsi členi zaporedja bodo torej cela števila z intervala $[-10^4, 10^4]$.

Izhod

Na izhodu izpišite niz `RAZLICNI` oziroma število, ki se v zaporedju največkrat ponovi.

Testni primer 1

Vhod:

```
12345 10
```

Izhod:

```
RAZLICNI
```

V tem primeru se tvori sledeče zaporedje:

```
-2595 3978 -1933 359 -9393 8040 -3678 -583 8214 9596
```

3.8 Izštevanka

Naloga

Otroci so razporejeni v ravni vrsti in se igrajo izštevanko tako dolgo, dokler v vrsti ne ostane samo eden. V vsakem krogu izštevanke izpade po en otrok, ki se določi na sledeči način: »Selektor«, ki ne pripada otrokom v vrsti, izgovarja besede izštevanke (npr. an-ban-pet-podgan-...) in istočasno s prstom »potuje« po otrocih. Prične pri prvem otroku v vrsti, po vsaki besedi pa pokaže na naslednjega otroka. Kadarkoli prispe do konca vrste, nadaljuje spet pri prvem otroku. Ko izgovori zadnjo besedo izštevanke, otrok, na katerega tedaj kaže njegov prst, izpade.

Na primer, če se na začetku v vrsti nahajajo otroci Ana, Bojan, Cene, Denis in Eva, izštevanka pa je sestavljena iz 9 besed, potem v prvem krogu izpade Denis:

Otroci	Ana	Bojan	Cene	Denis	Eva
Beseda izštevanke	1.	2.	3.	4.	5.
	6.	7.	8.	9.	

V igri ostanejo Ana, Bojan, Cene in Eva. Drugi krog se vnovič prične pri Ani (vsak krog se prične pri prvem otroku v vrsti!), konča pa se tudi pri Ani. Ostanejo torej Bojan, Cene in Eva. Tretji krog se prične pri Bojanu in konča pri Evi. Ostaneta Bojan in Cene. V četrtem krogu izpade Bojan. Sedaj se igra konča, saj je v vrsti ostal samo še en otrok (Cene).

Napišite program, ki prebere zaporedje imen otrok in število besed izštevanke, nato pa po vrsti izpisuje, kateri otroci izpadejo v posameznih krogih.

Vhod

V prvi vrstici vhoda je podano število otrok ($n \in [1, 100]$) in število besed izštevanke ($b \in [1, 10^9]$). Števili sta ločeni s presledkom. V drugi vrstici je podanih n imen otrok, ločenih s presledki. Imena berite z metodo `sc.next()`, kjer je `sc` objekt razreda `Scanner`.

Izhod

Izpišite $n - 1$ vrstic. V i-ti vrstici izpišite ime otroka, ki izpade v i-tem krogu izštevanke.

Testni primer

Vhod:

```
5 9
Ana Bojan Cene Denis Eva
```

Izhod:

```
Denis
Ana
Eva
Bojan
```

3.9 Izstopajoči element

Naloga

Dano je zaporedje najmanj treh celih števil, večjih od 1. Definirajmo pojem *izstopajočega elementa* zaporedja na sledeči način: element x *izstopa*, če je GCD (največji skupni delitelj) vseh ostalih elementov v zaporedju večji od 1, sam element x pa s tem GCD-jem ni deljiv. Na primer, v zaporedju $\{25, 40, 15, 36, 30\}$ izstopa element 36, saj je GCD ostalih elementov (25, 40, 15 in 30) enak 5, element 36 pa s tem GCD-jem ni deljiv. Napišite program, ki prebere zaporedje in po vrsti izpiše vse izstopajoče elemente v zaporedju. Če takih elementov ni, naj program izpiše `NIC`.

Vhod

V prvi vrstici je podano število $n \in [3, 10^3]$, v drugi pa n celih števil z intervala $[2, 10^9]$, med seboj ločenih s po enim presledkom.

Izhod

Na izhodu izpišite vse izstopajoče elemente v istem vrstnem redu, kot nastopajo v zaporedju. Vsak izstopajoči element izpišite v svoji vrstici. Če tovrstnih elementov ni, izpišite `NIC`.

Testni primer 1

Vhod:

```
7
24 60 36 18 54 40 48
```

Izhod:

```
40
```

Testni primer 2

Vhod:

```
7
24 60 36 18 54 42 48
```

Izhod:

```
NIC
```

Testni primer 3

Vhod:

```
3
15 20 18
```

Izhod:

```
15
20
18
```

V tem primeru vsi trije elementi izstopajo.

3.10 Kombinacije (⋆)

Naloga

Napišite program, ki prebere števili n in k in izpiše vsa strogo naraščajoča zaporedja k števil med 1 in n.

Vhod

Na vhodu sta podani celi števili $n \in [1, 15]$ in $k \in [1, n]$, ločeni s presledkom.

Izhod

Izpišite vsa iskana zaporedja, vsako v svoji vrstici. Zaporedja izpišite v leksikografskem vrstnem redu: najprej naraščajoče po prvem členu, nato (v okviru skupine zaporedij z isto vrednostjo prvega člena) naraščajoče po drugem členu itd. Vsako zaporedje naj bo izpisano v obliki, kot jo proizvede metoda `Arrays.toString`.

Testni primer 1

Vhod:

```
5 3
```

Izhod:

```
[1, 2, 3]
[1, 2, 4]
[1, 2, 5]
[1, 3, 4]
[1, 3, 5]
[1, 4, 5]
[2, 3, 4]
[2, 3, 5]
[2, 4, 5]
[3, 4, 5]
```

Napotek

Možnih je več pristopov. Pri enem od njih si pomagamo z rekurzivno metodo. Na i-tem nivoju rekurzije obravnavamo vsa možna števila na i-tem mestu v zaporedju. Na prvem nivoju tako obravnavamo vsa števila med 1 in n, na i-tem nivoju (pri $i \geqslant 2$) pa vsa števila, večja od trenutno izbranega števila na mestu $i - 1$. Za vsako izbrano število na i-tem nivoju rekurzivno obravnavamo vsa števila na nivoju $i + 1$. Na zadnjem nivoju rekurzije izpišemo zaporedje, ki smo ga pravkar dokončali.

3.11 Politična nasprotja I (⋆)

Naloga

Na politično konferenco je povabljenih l levičarjev, d desničarjev in c centristov. Organizatorji jih morajo razmestiti na $l + d + c$ zaporedno postavljenih sedežev, in to tako, da

levičar in desničar nikjer ne bosta soseda. Napišite program, ki prebere števila l, d in c in izpiše število vseh sedežnih redov, ki ustrezajo opisanemu pogoju.

Vhod

Na vhodu so podana cela števila $l \in [0, 10]$, $d \in [0, 10]$ in $c \in [0, 5]$, ločena s presledkom.

Izhod

Izpišite samo število možnih razporeditev. To število bo zagotovo manjše od 10^9.

Testni primer 1

Vhod:

```
2 3 2
```

Izhod:

```
15
```

V tem primeru so možne sledeče razporeditve:

```
LLCDDDC
LLCDDCD
LLCDCDD
LLCCDDD
LCLCDDD
LCDDDCL
DDDCLLC
DDDCLCL
DDDCCLL
DDCLLCD
DDCDCLL
DCLLCDD
DCDDCLL
CLLCDDD
CDDDCLL
```

Testni primer 2

Vhod:

```
2 3 1
```

Izhod:

```
2
```

V tem primeru sta možni le razporeditvi `LLCDDD` in `DDDCLL`.

3.12 Štetje klicev II (⋆)

Naloga

Podani sta celi števili $a \geqslant 2$ in $b \geqslant 2$. Funkcija $f\colon \mathbb{N}_0 \to \mathbb{N}_0$ je definirana takole:

$$f(n) = \begin{cases} 1 & \text{pri } n = 0 \\ f(\lfloor \frac{n}{a} \rfloor) + f(\lfloor \frac{n}{b} \rfloor) & \text{pri } n > 0 \end{cases}$$

Zapis $\lfloor r \rfloor$ označuje celi del realnega števila r (npr. $\lfloor 2{,}8 \rfloor = 2$).

Napišite program, ki prebere števila a, b in n in izpiše število klicev funkcije f, če vrednost $f(n)$ izračunamo strogo po definiciji, pri čemer večkratne klice z istim parametrom štejemo samo po enkrat.

Vhod

Na vhodu so podana cela števila $a \in [2,\ 100]$, $b \in [2,\ 100]$ in $n \in [0,\ 10^6]$, ločena s presledkom.

Izhod

Izpišite število različnih klicev funkcije. To število bo zanesljivo manjše od 10^9.

Testni primer 1

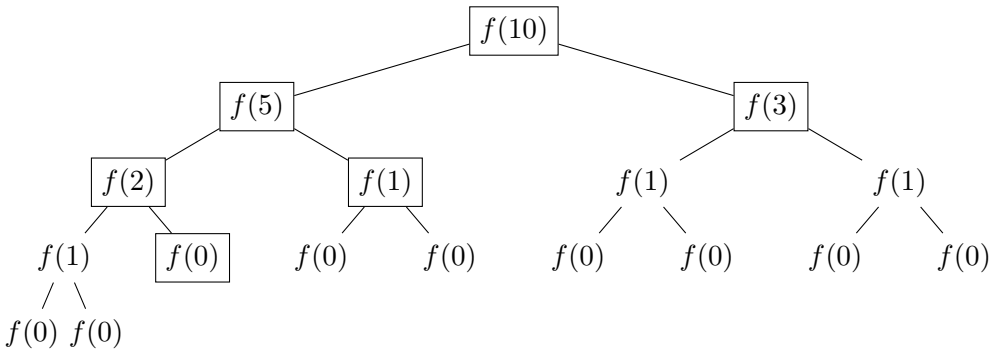
Vhod:

```
2 3 10
```

Izhod:

```
6
```

V tem primeru imamo 6 različnih klicev funkcije f:



3.13 Maksimumi po stolpcih I

Naloga

Napišite program, ki prebere celoštevilsko matriko in izpiše maksimalne elemente po posameznih stolpcih.

Vhod

V prvi vrstici sta podani celi števili $v \in [1, 100]$ in $s \in [1, 100]$, nato pa sledi še v vrstic. V vsaki od njih je zapisanih po s celih števil z intervala $[-10^9, 10^9]$, ki tvorijo vsebino pripadajoče vrstice matrike.

Izhod

Izpišite maksimalne elemente posameznih stolpcev matrike, in sicer v obliki, kot jo proizvede metoda `Arrays.toString`.

Testni primer 1

Vhod:

```
5 4
10 -6 -5 1
0 -1 -7 -3
9 5 -4 7
8 3 -3 7
4 2 -6 9
```

Izhod:

```
[10, 5, -3, 9]
```

3.14 Maksimumi po stolpcih II

Naloga

Napišite program, ki prebere **ne nujno pravokotno** celoštevilsko matriko in izpiše maksimalne elemente po posameznih stolpcih. Število izpisanih elementov naj bo enako dolžini najdaljše vrstice matrike.

Vhod

V prvi vrstici je podano celo število $n \in [1, 100]$, nato pa sledi še n vrstic. Vsaka od njih se prične s celim številom $d \in [1, 100]$, zatem pa sledi d celih števil z intervala $[-10^9, 10^9]$, ki tvorijo vsebino pripadajoče vrstice matrike.

Izhod

Izpišite maksimalne elemente posameznih stolpcev matrike, in sicer v obliki, kot jo proizvede metoda `Arrays.toString`.

Testni primer 1

Vhod:

```
5
5 9 5 3 -2 -4
2 3 -4
5 10 2 -5 -9 8
6 -5 6 -2 -7 -3 -4
4 6 -2 6 -3
```

Izhod:

```
[10, 6, 6, -2, 8, -4]
```

V tem primeru vsebuje prvi stolpec matrike števila 9, 3, 10, -5 in 6, drugi števila 5, -4, 2, 6 in -2, ..., zadnji pa samo število -4.

3.15 Pravilni trikotniki

Naloga

Napišite program, ki prebere zaporedje parov celoštevilskih koordinat ravninskih točk in poišče vse trojice točk, ki tvorijo pravilne trikotnike v okviru določene tolerance. Trikotnik proglasimo za pravilnega, če je razlika med dolžino njegove najdaljše stranice in dolžino njegove najkrajše stranice manjša od (10^{-d})-kratnika dolžine njegove najkrajše stranice, kjer je d neko pozitivno celo število. Če pravilnih trikotnikov ni, naj program to sporoči.

Vhod

V prvi vrstici sta podani celi števili $d \in [1, 10]$ in $n \in [1, 100]$, nato pa sledi n vrstic, ki podajajo koordinate posameznih točk. V vsaki vrstici sta zapisani celi števili z intervala $[-2 \cdot 10^4, 2 \cdot 10^4]$, ki po vrsti predstavljata koordinati x in y.

Izhod

Izpišite vse trojice indeksov (i, j in k) točk, ki tvorijo pravilne trikotnike. Vsaka trojica naj se izpiše samo enkrat, in sicer v obliki

$i␣j␣k$

pri čemer velja $i < j < k$. Trojice izpišite v leksikografskem vrstnem redu (najprej naraščajoče po indeksih i, nato po indeksih j, nazadnje pa po indeksih k).

Če nobena trojica ne tvori pravilnega trikotnika, naj program izpiše `BREZ`.

Testni primer 1

Vhod:

```
2 10
100 100
-23 287
473 373
300 200
163 163
437 237
400 100
250 360
337 337
200 300
```

Izhod:

```
0 1 9
0 6 7
2 5 8
```

```
2 6 9
3 4 9
3 5 6
3 5 8
3 8 9
4 6 8
```

Testni primer 2

Vhod:

```
2 4
10 10
30 20
50 70
10 80
```

Izhod:

```
BREZ
```

3.16 Leksikografsko urejanje

Naloga

Vektor $(a_0, a_1, \ldots, a_{n-1})$ je *leksikografsko manjši* od vektorja $(b_0, b_1, \ldots, b_{n-1})$, če obstaja indeks $i \geqslant 0$, tako da velja (1) $a_i < b_i$ in (2) $a_j = b_j$ za vsak $j < i$. Na primer, vektor $a = (5, 3, 4)$ je leksikografsko manjši od vektorjev $b = (5, 3, 8)$ ($a_2 < b_2$, $a_1 = b_1$, $a_0 = b_0$), $c = (5, 6, 1)$ ($a_1 < c_1$, $a_0 = c_0$) in $d = (6, 2, 7)$ ($a_0 < d_0$), ne pa od vektorja $e = (3, 6, 7)$. Napišite program, ki prebere n vektorjev dolžine d, nato pa jih izpiše v leksikografskem vrstnem redu.

Opomba: Povsem enak način urejanja se uporablja pri nizih (npr. pri priimkih, slovarskih geslih itd.), le da tam v vlogi vektorjev nastopajo nizi, v vlogi posameznih elementov vektorjev pa znaki.

Vhod

V prvi vrstici sta zapisani celi števili $n \in [1, 100]$ in $d \in [1, 100]$. Nato sledi n vrstic, ki podajajo vsebino posameznih vektorjev. Vsaka od njih vsebuje po d celih števil z intervala $[-10^9, 10^9]$.

Izhod

Izpišite iste vektorje, urejene v leksikografskem vrstnem redu. Vsak vektor izpišite v svoji vrstici, in to v obliki, kot jo proizvede metoda `Arrays.toString`.

Testni primer 1

Vhod:

```
5 3
6 2 7
3 6 7
```

```
5 6 1
5 3 4
5 3 8
```

Izhod:

```
[3, 6, 7]
[5, 3, 4]
[5, 3, 8]
[5, 6, 1]
[6, 2, 7]
```

Testni primer 2

Vhod:

```
7 5
-3 5 -7 2 -6
-3 4 -2 8 -4
-3 4 -2 8 -9
-5 10 6 10 5
-3 4 2 8 -9
-3 4 -2 5 -4
-3 4 -2 8 -4
```

Izhod:

```
[-5, 10, 6, 10, 5]
[-3, 4, -2, 5, -4]
[-3, 4, -2, 8, -9]
[-3, 4, -2, 8, -4]
[-3, 4, -2, 8, -4]
[-3, 4, 2, 8, -9]
[-3, 5, -7, 2, -6]
```

3.17 Šahovski turnir

Naloga

Na šahovskem turnirju nastopa n igralcev. Napišite program, ki prebere rezultate posameznih partij (v obliki »igralec A, ki je vodil bele figure, je proti igralcu B, ki je vodil črne figure, zmagal/izgubil/remiziral«) in izpiše turnirsko lestvico. Turnirska lestvica je seznam igralcev, padajoče urejen po skupnem številu točk. Za potrebe te naloge privzemimo, da vsaka zmaga prinese po 2 točki, remi po 1 točko, poraz pa po 0 točk. (Dejansko šahovsko točkovanje je $1/\frac{1}{2}/0$, vendar pa bi se radi izognili decimalkam.)

Vhod

V prvi vrstici je podano celo število $n \in [2, 100]$, nato pa sledi vnaprej neznano število vrstic, ki podajajo rezultate posameznih partij. Vsaka vrstica je sledeče oblike:

štBelega␣*štČrnega*␣*izid*

Pri tem je *štBelega* zaporedna številka igralca z belimi figurami, *štČrnega* zaporedna številka igralca s črnimi figurami, *izid* pa je enak 1 (zmagal je beli), -1 (zmagal je črni) ali 0 (remi). Zaporedne številke so seveda cela števila z intervala $[1, n]$, velja pa tudi *štBelega* $\neq$ *štČrnega*.

Izhod

Izpišite n vrstic sledeče oblike:

zapŠtIgralca␣*točke*

Pri tem je *zapŠtIgralca* zaporedna številka igralca, *točke* pa njegova skupna vsota točk. Zaporedje vrstic naj bo urejeno po padajočih točkah, v primeru enakega števila točk pa po naraščajočih zaporednih številkah.

Testni primer 1

Vhod:

```
5
3 2 1
2 4 0
2 4 -1
5 1 -1
3 5 0
1 3 -1
5 2 1
3 1 -1
```

Izhod:

```
3 5
1 4
4 3
5 3
2 1
```

3.18 Determinanta

Naloga

Napišite program, ki prebere kvadratno celoštevilsko matriko in izračuna njeno determinanto po sledeči definiciji (za $n > 1$):

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{34} & \dots & a_{3n} \\ a_{41} & a_{42} & a_{43} & a_{44} & \dots & a_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \dots & a_{nn} \end{vmatrix} = (-1)^0 \, a_{11} \begin{vmatrix} a_{22} & a_{23} & a_{24} & \dots & a_{2n} \\ a_{32} & a_{33} & a_{34} & \dots & a_{3n} \\ a_{42} & a_{43} & a_{44} & \dots & a_{4n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n2} & a_{n3} & a_{n4} & \dots & a_{nn} \end{vmatrix}$$

$$+ (-1)^1 \, a_{12} \begin{vmatrix} a_{21} & a_{23} & a_{24} & \dots & a_{2n} \\ a_{31} & a_{33} & a_{34} & \dots & a_{3n} \\ a_{41} & a_{43} & a_{44} & \dots & a_{4n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n3} & a_{n4} & \dots & a_{nn} \end{vmatrix}$$

$$+ (-1)^2 \, a_{13} \begin{vmatrix} a_{21} & a_{22} & a_{24} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{34} & \dots & a_{3n} \\ a_{41} & a_{42} & a_{44} & \dots & a_{4n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n4} & \dots & a_{nn} \end{vmatrix}$$

$$+ \dots$$

$$+ (-1)^{n-1} \, a_{1n} \begin{vmatrix} a_{21} & a_{22} & a_{23} & \dots & a_{2,\,n-1} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3,\,n-1} \\ a_{41} & a_{42} & a_{43} & \dots & a_{4,\,n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{n,\,n-1} \end{vmatrix}$$

Determinanta matrike velikosti 1×1 je kar enaka edinemu elementu te matrike.

Opomba: Determinante v praksi ne računamo po gornji definiciji, saj vodi do neučinkovite in numerično nestabilne kode. Z vidika učenja programiranja pa je ta definicija odlična, zato jo v vašem programu striktno upoštevajte.

Vhod

V prvi vrstici je podano celo število $n \in [1, 8]$, nato pa sledi n vrstic. V vsaki od njih je zapisanih po n celih števil z intervala $[-100, 100]$, ki podajajo vsebino pripadajoče vrstice matrike.

Izhod

Izpišite samo determinanto. Determinanta bo po absolutni vrednosti zanesljivo manjša od 10^9.

Testni primer 3

Vhod:

```
3
3 4 -2
5 7 8
-3 10 5
```

Izhod:

```
-473
```

3.19 Politična nasprotja II (⋆)

Naloga

Opomba: Ta naloga je enaka nalogi *Politična nasprotja* iz sklopa dodatnih nalog za 5. teden predavanj, le vhodni podatki so lahko večji.

Na politično konferenco je povabljenih l levičarjev, d desničarjev in c centristov. Organizatorji jih morajo razmestiti na $l + d + c$ zaporedno postavljenih sedežev, in to tako, da levičar in desničar nikjer ne bosta soseda. Napišite program, ki prebere števila l, d in c in izpiše število vseh sedežnih redov, ki ustrezajo opisanemu pogoju.

Vhod

Na vhodu so podana cela števila $l \in [0, 20]$, $d \in [0, 20]$ in $c \in [0, 20]$, ločena s presledkom.

Izhod

Izpišite samo število možnih razporeditev. To število bo zagotovo manjše od 10^{18}.

Testni primer 1

Vhod:

```
2 3 2
```

Izhod:

```
15
```

V tem primeru so možne sledeče razporeditve:

```
LLCDDDC
LLCDDCD
LLCDCDD
LLCCDDD
LCLCDDD
LCDDDCL
DDDCLLC
DDDCLCL
DDDCCLL
DDCLLCD
DDCDCLL
DCLLCDD
DCDDCLL
CLLCDDD
CDDDCLL
```

Testni primer 2

Vhod:

```
2 3 1
```

Izhod:

```
2
```

V tem primeru sta možni le razporeditvi `LLCDDD` in `DDDCLL`.

4 Razredi in objekti

4.1 Razreda `Posta` in `Pismo`

Naloga

V skladu s sledečimi navodili napišite razreda `Posta` in `Pismo`.

Razred `Posta`

Razred `Posta` definirajte tako, da bo vsak njegov objekt predstavljal neko pošto s poštno številko (npr. 1000) in nazivom (npr. Ljubljana). Razred naj vsebuje sledeče konstruktorje in metode:

- `public Posta(int stevilka, String naziv)`

  Ustvari nov objekt tipa `Posta`, ki predstavlja pošto s podano poštno številko in nazivom.

- `public int vrniStevilko()`

  Vrne poštno številko pošte `this`.

- `public String vrniNaziv()`

  Vrne naziv pošte `this`.

- `public String toString()`

  Za pošto `this` vrne niz sledeče oblike:

  stevilka␣naziv

  Na primer:

  `1000␣Ljubljana`

Razred `Pismo`

Razred `Pismo` definirajte tako, da bo vsak njegov objekt predstavljal neko pismo s sledečimi podatki:

- izvorna pošta (npr. 1000 Ljubljana);
- ciljna pošta (npr. 2000 Maribor);

- podatek o tem, ali je pismo priporočeno ali navadno;
- razdalja (v kilometrih) med izvorno in ciljno pošto.

V razredu definirajte sledeče javno dostopne konstruktorje in metode:

- `public Pismo(Posta izvorna, Posta ciljna,`
  `boolean jePriporoceno, int razdalja)`

  Ustvari nov objekt tipa `Pismo`, ki predstavlja pismo s podano izvorno in ciljno pošto, »priporočenostjo« (`true`: priporočeno; `false`: navadno) ter razdaljo (v kilometrih) med izvorno in ciljno pošto.

- `public String toString()`

  Za pismo `this` vrne niz oblike

  izvorna_pošta␣`->`␣*ciljna_pošta*␣`(`*razdalja*␣`km)`␣`[`*vrsta*`]`

  pri čemer je *vrsta* bodisi `P` (priporočeno) ali pa `N` (navadno). Na primer:

  `1000`␣`Ljubljana`␣`->`␣`2000`␣`Maribor`␣`(130`␣`km)`␣`[P]`

- `public boolean izviraOd(Posta posta)`

  Vrne `true` natanko v primeru, če je pošta `posta` izvorna pošta za pismo `this`.

- `public boolean staIzvorInCiljIsta()`

  Vrne `true` natanko v primeru, če ima pismo `this` isto izvorno in ciljno pošto (npr. če je pismo poslano s pošte 1000 Ljubljana na pošto 1000 Ljubljana).

- `public boolean imaIstiCiljKot(Pismo pismo)`

  Vrne `true` natanko v primeru, če ima pismo `this` isto ciljno pošto kot pismo `pismo`.

- `public static boolean imataIstiCilj(Pismo p1, Pismo p2)`

  Vrne `true` natanko v primeru, če imata obe podani pismi isto ciljno pošto.

- `public int cena()`

  Vrne ceno (v stotinih) oddaje pisma `this`. Za navadno pismo se cena izračuna glede na razdaljo: za razdaljo od 0 do vključno $(r-1)$ km je cena enaka c stotinov, za razdaljo od r do vključno $(2r-1)$ km znaša $2c$ stotinov, za razdaljo od $2r$ do vključno $(3r-1)$ km znaša $3c$ stotinov itd. Ceno priporočenega pisma izračunamo tako, da ceni navadnega pisma prištejemo priporočnino p stotinov, ki je neodvisna od razdalje. Konstante r, c in p se nastavijo z metodo, ki jo predstavljamo v naslednji alineji.

- `public static void nastaviKonstanteZaCeno(`
  `int enotaRazdalje, int enotaCene, int priporocnina)`

  Konstante r, c in p, ki se uporabljajo za izračun cene oddaje pisma (gl. prejšnjo alinejo), nastavi na vrednosti `enotaRazdalje`, `enotaCene` in `priporocnina` (v tem vrstnem redu).

- `public boolean jeDrazjeOd(Pismo pismo)`

  Vrne `true` natanko v primeru, če je cena pisma `this` večja od cene pisma `pismo`.

- `public static Pismo vrniDrazje(Pismo p1, Pismo p2)`

Vrne tisto pismo izmed p1 in p2, ki ima večjo ceno. Če imata obe pismi enako ceno, naj vrne pismo p2.

- public Pismo izdelajPovratno()

  Ustvari in vrne nov objekt tipa Pismo, ki predstavlja povratnico pisma this. Povratnica ima enake podatke kot pismo this, le izvorna in ciljna pošta sta med seboj zamenjani.

Testni primer 11

Testni razred:

```
public class Test11 {

    public static void main(String[] args) {
        Posta lj = new Posta(1000, "Ljubljana");
        Posta mb = new Posta(2000, "Maribor");
        Posta ce = new Posta(3000, "Celje");
        System.out.println(lj.vrniNaziv());
        System.out.println(mb.vrniStevilko());
        System.out.println(ce.toString());

        Pismo.nastaviKonstanteZaCeno(10, 3, 20);
        Pismo lj2ce = new Pismo(lj, ce, true, 75);
        Pismo mb2lj = new Pismo(mb, lj, false, 130);
        Pismo ce2ce = new Pismo(ce, ce, true, 0);
        System.out.println(lj2ce.izviraOd(mb));
        System.out.println(ce2ce.staIzvorInCiljIsta());
        System.out.println(lj2ce.imaIstiCiljKot(ce2ce));
        System.out.println(Pismo.imataIstiCilj(lj2ce, mb2lj));
        System.out.println(lj2ce.cena());
        System.out.println(mb2lj.cena());
        System.out.println(mb2lj.jeDrazjeOd(lj2ce));
        System.out.println(Pismo.vrniDrazje(mb2lj, lj2ce).toString());
        System.out.println(mb2lj.izdelajPovratno().toString());
    }
}
```

Izhod:

```
Ljubljana
2000
3000 Celje
false
true
true
false
44
42
false
1000 Ljubljana -> 3000 Celje (75 km) [P]
1000 Ljubljana -> 2000 Maribor (130 km) [N]
```

4.2 Razred `Ulomek`

Naloga

Napišite razred `Ulomek` tako, da bodo njegovi objekti predstavljali posamezne okrajšane ulomke. Razred `Ulomek` naj vsebuje sledeče konstruktorje in metode:

- `public Ulomek(int a, int b)`

  Ustvari nov objekt tipa `Ulomek`, ki predstavlja okrajšano različico ulomka `a`/`b`. Ulomek p/q je okrajšan natanko tedaj, ko velja $q > 0$ in $\gcd(p, q) = 1$. Na primer, okrajšana različica ulomka 15/5 je ulomek 3/1, okrajšana različica ulomka $10/(-20)$ pa je ulomek $-1/2$. Lahko predpostavite, da sta števili `a` in `b` različni od 0.

- `public String toString()`

  Vrne okrajšani ulomek `this` v obliki niza *števec*/*imenovalec*, npr. `3/1` ali `-1/2`.

- `public boolean jeEnakKot(Ulomek u)`

  Vrne `true` natanko v primeru, če sta ulomka `this` in `u` enaka.

- `public Ulomek negacija()`

  Ustvari in vrne nov objekt, ki predstavlja nasprotno vrednost ulomka `this` (torej $-x$, če je x dani ulomek).

- `public Ulomek obrat()`

  Ustvari in vrne nov objekt, ki predstavlja obratno vrednost ulomka `this` (torej $1/x$, če je x dani ulomek).

- `public Ulomek vsota(Ulomek u)`
  `public Ulomek razlika(Ulomek u)`
  `public Ulomek zmnozek(Ulomek u)`
  `public Ulomek kolicnik(Ulomek u)`

  Ustvari in vrne nov objekt, ki predstavlja vsoto, razliko, zmnožek oziroma količnik ulomka `this` in ulomka `u`.

- `public Ulomek potenca(int eksponent)`

  Vrne potenco ulomka `this` na podani eksponent. Eksponent ni nujno pozitiven!

- `public boolean jeManjsiOd(Ulomek u)`

  Vrne `true` natanko v primeru, če je ulomek `this` manjši od ulomka `u`.

Testni primer 11

Testni razred:

```
public class Test11 {

    public static void main(String[] args) {
        Ulomek a = new Ulomek(-30, -40);
        Ulomek b = new Ulomek(24, -18);
        Ulomek c = new Ulomek(15, 20);

        System.out.println(a.toString());
        System.out.println(a.jeEnakKot(c));
```

```
        System.out.println(a.negacija().toString());
        System.out.println(a.obrat().toString());
        System.out.println(a.vsota(b).toString());
        System.out.println(a.razlika(b).toString());
        System.out.println(a.zmnozek(b).toString());
        System.out.println(a.kolicnik(b).toString());
        System.out.println(a.potenca(2).toString());
        System.out.println(b.potenca(-3).toString());
        System.out.println(a.jeManjsiOd(b));
    }
}
```

Izhod:

```
3/4
true
-3/4
4/3
-7/12
25/12
-1/1
-9/16
9/16
-27/64
false
```

Opomba

Metode, ki jih morate realizirati, so zasnovane tako, da ne spreminjajo stanja objekta `this`. Če boste to načelo uporabili za vse metode vašega razreda `Ulomek`, bodo objekti tipa `Ulomek` *nespremenljivi* (angl. immutable). Stanja nespremenljivega objekta po izdelavi ne moremo več spreminjati. Nespremenljivi objekti imajo vrsto dobrih lastnosti in lahko programerju prihranijo marsikatero skrb:

`http://www.javapractices.com/topic/TopicAction.do?Id=29`

Na primer, sledeči potencialno zahrbtni pojavi so možni zgolj pri spremenljivih objektih, saj nespremenljivi po definiciji sploh ne morejo imeti metod vrste »setter«:

```
Oseba os1 = new Oseba("Jože", "Gorišek", "Ruše");    // ime, priimek, kraj
Oseba os2 = os1;
os1.nastaviKraj("Maribor");
os1.izpisi();   // Jože Gorišek, Maribor
os2.izpisi();   // Jože Gorišek, Maribor (past!)
```

4.3 Razred `Datum`

Naloga

Napišite razred `Datum` tako, da bodo njegovi objekti predstavljali veljavne datume po gregorijanskem koledarju. Hraniti želimo datume od 1. januarja 1583 (torej od leta, ko se je uveljavil gregorijanski koledar) do 31. decembra 2999. Razred `Datum` naj vsebuje sledeče metode:

- `public static Datum ustvari(int dan, int mesec, int leto)`

  Če podani parametri predstavljajo veljaven datum (`leto` med 1583 in 2999, `mesec` od 1 do 12, `dan` od 1 do števila dni v izbranem mesecu izbranega leta), naj metoda ustvari in vrne nov objekt razreda `Datum`, sicer pa naj vrne `null`. Pri preverjanju veljavnosti datuma upoštevajte pravilo za prestopna leta, ki se glasi takole: leto je prestopno, če je deljivo s 400 ali pa če je deljivo s 4, vendar ni deljivo s 100. Leta 1700, 1800, 1900, 2100, 2200, 2300, 2500, ... tako niso prestopna, leta 1600, 2000, 2400, ... pa so.

  Zakaj smo se v tem primeru odločili za statično konstrukcijsko metodo namesto za konstruktor? Ali je konstruktor kljub tej metodi smiselno napisati? Če da, naj bo javno dostopen ali privaten?

- `public String toString()`

  Vrne predstavitev datuma v obliki niza `DD.MM.LLLL`, npr. `15.07.2014` ali `05.11.1975`.

- `public boolean jeEnakKot(Datum datum)`

  Vrne `true` natanko v primeru, če objekt `this` predstavlja isti datum kot objekt `datum`.

- `public boolean jePred(Datum datum)`

  Vrne `true` natanko v primeru, če je datum `this` kronološko pred datumom `datum`. Na primer, datum 30.09.2014 je pred datumom 27.10.2014, ta pa je pred datumom 20.01.2015.

- `public Datum naslednik()`

  Ustvari in vrne nov objekt razreda `Datum`, ki predstavlja neposrednega naslednika datuma `this`. Na primer, naslednik datuma 28.02.2012 je datum 29.02.2012, njegov naslednik pa je datum 01.03.2012. Naslednik datuma 28.02.2014 je datum 01.03.2014. Če datum nima veljavnega naslednika (edini tak datum je 31.12.2999), naj metoda vrne `null`.

- `public Datum predhodnik()`

  Ustvari in vrne nov objekt razreda `Datum`, ki predstavlja neposrednega predhodnika datuma `this`. Na primer, predhodnik datuma 01.03.2012 je datum 29.02.2012, predhodnik datuma 01.03.2014 pa je datum 28.03.2014. Če datum nima veljavnega predhodnika (edini tak datum je 01.01.1583), naj metoda vrne `null`.

- `public Datum cez(int stDni)`

  Izračuna (in vrne kot nov objekt tipa `Datum`) datum, ki je za `stDni` oddaljen od datuma `this`. Parameter `stDni` je lahko tudi negativen; v tem primeru metoda izračuna datum, ki je `-stDni` pred datumom `this`. Če ciljni datum pade pred datum 01.01.1583 ali za datum 31.12.2999, naj metoda vrne `null`.

- `public int razlika(Datum datum)`

  Vrne razliko (v številu dni) med datumoma `this` in `datum`. Če je datum `this` pred datumom `datum`, je razlika seveda negativna.

Testni primer 11

Testni razred:

```
public class Test11 {

    public static void main(String[] args) {
        Datum a = Datum.ustvari(29, 2, 2016);
        Datum b = Datum.ustvari(1, 1, 2017);

        System.out.println(a.toString());
        System.out.println(a.jeEnakKot(b));
        System.out.println(a.jePred(b));
        System.out.println(a.naslednik().toString());
        System.out.println(b.predhodnik().toString());
        System.out.println(a.cez(365).toString());
        System.out.println(b.cez(-365).toString());
    }
}
```

Izhod:

```
29.02.2016
false
true
01.03.2016
31.12.2016
28.02.2017
02.01.2016
```

Opomba

Statična konstrukcijska metoda (`ustvari` v našem primeru) se imenuje *tovarna* (angl. factory method). Tovarne so namenjene nadzorovani izdelavi objektov in se zato uporabljajo v kombinaciji s privatnimi konstruktorji. V tej nalogi smo tovarno izdelali zato, ker želimo zagotoviti, da objekti razreda `Datum` predstavljajo samo veljavne datume. Samo s konstruktorjem tega ne bi mogli doseči, saj konstruktor ne more preprečiti izdelave objekta.

4.4 Dopolnitve razreda `Oseba` (⋆)

Naloga

V razred `Oseba`, ki smo ga napisali na vajah in ki ga najdete tudi v mapi s testnimi primeri, dodajte sledeče metode:

- `public int ocetovskaGeneracijskaRazlika(Oseba os)`

  Vrne očetovsko generacijsko razliko med osebama `this` in `os` (OGR(`this`, `os`)). Vrednost $OGR(A, B)$ za osebi A in B je definirana takole:

  - Če sta osebi A in B identični (če gre za isto osebo), velja $OGR(A, B) = 0$.
  - Če je oseba A očetovski prednik osebe B, potem $OGR(A, B)$ izračunamo kot število očetov na liniji od B do A. (Če je oseba A oče osebe B, velja $OGR(A, B) = 1$, če je oseba A oče očeta osebe B, velja $OGR(A, B) = 2$ itd.)
  - Če je oseba B očetovski prednik osebe A, velja $OGR(A, B) = -OGR(B, A)$.

  - Če nobeden od zgornjih pogojev ni izpolnjen, naj metoda namesto vračila vrednosti (torej namesto stavka `return`) sproži izjemo tipa `IllegalArgumentException`:

    ```
    throw new IllegalArgumentException();
    ```

- `public boolean jePrednikOd(Oseba os)`

  Vrne `true` natanko v primeru, če je oseba `this` prednik osebe `os`. Oseba A je prednik osebe B, če je izpolnjen eden od sledečih pogojev:

  - $A = B$ (vsaka oseba je sam svoj prednik).
  - A je prednik B-jevega očeta.
  - A je prednik B-jeve matere.

- `public void nastejPrednike()`

  Izpiše vse prednike osebe `this`. Vsak prednik naj se izpiše v svoji vrstici, in sicer na sledeči način:

  veriga`:␣`*prednik*

  Pri tem je *veriga* niz, ki ponazarja starševsko verigo od osebe `this` do prednika, izpisanega v tekoči vrstici, *prednik* pa niz, ki ga za danega prednika vrne metoda `toString`. Niz *veriga* izdelajte po sledečem zgledu: niz `this` predstavlja osebo `this`, niz `this.oce` predstavlja očeta osebe `this`, niz `this.oce.mati` predstavlja mater očeta osebe `this` itd. Najprej naj se na opisani način izpišejo vsi predniki po očetovi strani, nato pa še vsi predniki po materini strani. Enako pravilo naj se rekurzivno uporabi za posamezne prednike. Za primer s slike 4.1 bi klic `os43.nastejPrednike()` izpisal sledeče:

  ```
  this: Gal Smole [M] (2009)
  this.oce: Miha Smole [M] (1978)
  this.oce.oce: Milan Smole [M] (1953)
  this.oce.mati: Metka Smole [Z] (1953)
  this.mati: Neža Smole [Z] (1980)
  this.mati.oce: Zvone Kotnik [M] (1956)
  this.mati.oce.oce: France Kotnik [M] (1932)
  this.mati.oce.mati: Ivana Kotnik [Z] (1931)
  this.mati.mati: Tanja Kotnik [Z] (1954)
  ```

- `public boolean jeSorodnikOd(Oseba os)`

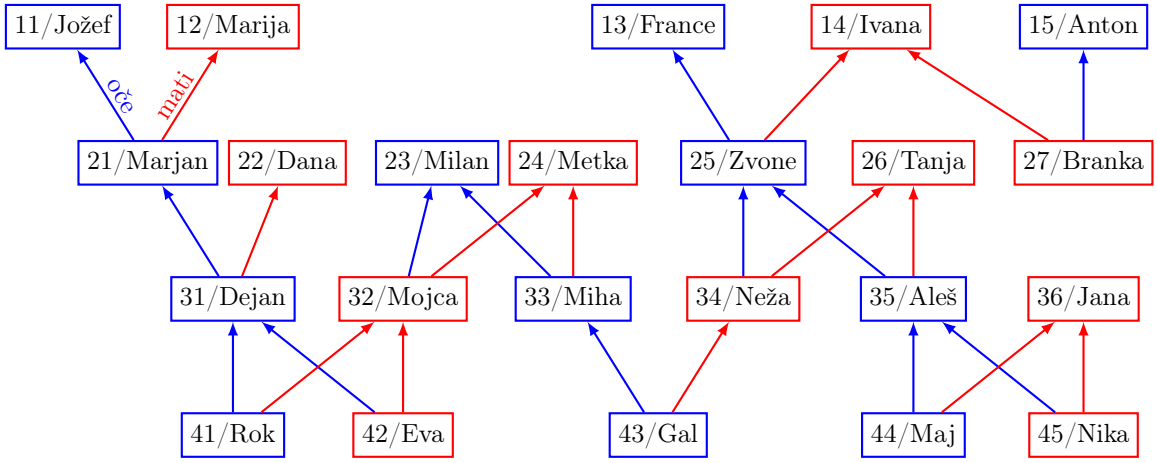
  Vrne `true` natanko v primeru, če sta osebi `this` in `os` sorodnika. Osebi A in B sta sorodnika, če obstaja oseba C, ki je prednik tako osebe A kot osebe B. (Tudi sorodstvo je možno definirati na eleganten rekurzivni način. Odkrijte ga sami!)

  V primeru z vaj (slika 4.1) sta osebi `os41` in `os43` (Rok in Gal) sorodnika. Sorodnika sta tudi osebi `os24` in `os42` (Metka in Eva), osebi `os33` in `os34` (Miha in Neža) pa nista.

Testni primer 11

Testni razred:

```
public class Test11 {
```



Slika 4.1: *Starševski odnosi med osebami v testnem razredu.*

```
    public static void main(String[] args) {
        Oseba os11 = new Oseba("Jožef", "Pogačnik", 'M', 1921, null, null);
        Oseba os12 = new Oseba("Marija", "Pogačnik", 'Z', 1928, null, null);
        Oseba os13 = new Oseba("France", "Kotnik", 'M', 1932, null, null);
        Oseba os14 = new Oseba("Ivana", "Kotnik", 'Z', 1931, null, null);
        Oseba os15 = new Oseba("Anton", "Zajc", 'M', 1922, null, null);
        Oseba os21 = new Oseba("Marjan", "Pogačnik", 'M', 1946, os11, os12);
        Oseba os22 = new Oseba("Dana", "Pogačnik", 'Z', 1950, null, null);
        Oseba os23 = new Oseba("Milan", "Smole", 'M', 1953, null, null);
        Oseba os24 = new Oseba("Metka", "Smole", 'Z', 1953, null, null);
        Oseba os25 = new Oseba("Zvone", "Kotnik", 'M', 1956, os13, os14);
        Oseba os26 = new Oseba("Tanja", "Kotnik", 'Z', 1954, null, null);
        Oseba os27 = new Oseba("Branka", "Zajc", 'Z', 1952, os15, os14);
        Oseba os31 = new Oseba("Dejan", "Pogačnik", 'M', 1973, os21, os22);
        Oseba os32 = new Oseba("Mojca", "Pogačnik", 'Z', 1977, os23, os24);
        Oseba os33 = new Oseba("Miha", "Smole", 'M', 1978, os23, os24);
        Oseba os34 = new Oseba("Neža", "Smole", 'Z', 1980, os25, os26);
        Oseba os35 = new Oseba("Aleš", "Kotnik", 'M', 1982, os25, os26);
        Oseba os36 = new Oseba("Jana", "Kotnik", 'Z', 1981, null, null);
        Oseba os41 = new Oseba("Rok", "Pogačnik", 'M', 2003, os31, os32);
        Oseba os42 = new Oseba("Eva", "Pogačnik", 'Z', 2006, os31, os32);
        Oseba os43 = new Oseba("Gal", "Smole", 'M', 2009, os33, os34);
        Oseba os44 = new Oseba("Maj", "Kotnik", 'M', 2010, os35, os36);
        Oseba os45 = new Oseba("Nika", "Kotnik", 'Z', 2012, os35, os36);

        System.out.println( os13.ocetovskaGeneracijskaRazlika(os45) );
        System.out.println( os42.ocetovskaGeneracijskaRazlika(os21) );
        System.out.println( os24.jePrednikOd(os42) );
        System.out.println( os14.jePrednikOd(os33) );
        os41.nastejPrednike();
        System.out.println( os41.jeSorodnikOd(os43) );
        System.out.println( os42.jeSorodnikOd(os45) );
        System.out.println( os13.jeSorodnikOd(os44) );
    }
}
```

Izhod:

```
3
-2
true
false
this: Rok Pogačnik (M), 2003
this.oce: Dejan Pogačnik (M), 1973
this.oce.oce: Marjan Pogačnik (M), 1946
this.oce.oce.oce: Jožef Pogačnik (M), 1921
this.oce.oce.mati: Marija Pogačnik (Z), 1928
this.oce.mati: Dana Pogačnik (Z), 1950
this.mati: Mojca Pogačnik (Z), 1977
this.mati.oce: Milan Smole (M), 1953
this.mati.mati: Metka Smole (Z), 1953
true
false
true
```

4.5 Tabela s poljubnim številom dimenzij (⋆)

Naloga

Napišite razred `Ptabela`, čigar objekt predstavlja neko (hiper-)pravokotno celoštevilsko tabelo s *poljubnim* številom dimenzij (»p-tabela«). Razred naj ponuja sledeče konstruktorje in metode:

- `public Ptabela(int[] dimenzije)`

  Ustvari p-tabelo s podanimi velikostmi dimenzij in jo napolni z ničlami. Lahko predpostavite, da je dolžina tabele `dimenzije` enaka najmanj 1.

  Na primer, stavek

  ```
  Ptabela p = new Ptabela(new int[]{3, 4, 2});
  ```

  bi ustvaril p-tabelo velikosti $3 \times 4 \times 2$.

- `public void nastavi(int[] indeksi, int vrednost)`

  Element p-tabele `this`, določen s podano tabelo indeksov, nastavi na podano vrednost. Lahko predpostavite, da je dolžina tabele `indeksi` enaka številu dimenzij p-tabele `this` in da so vsi indeksi veljavni.

  Na primer, stavek

  ```
  p.nastavi(new int[]{1, 3, 0}, 25);
  ```

  bi nastavil element na poziciji `[1][3][0]` (druga »stran«, četrta vrstica, prvi element znotraj vrstice) na vrednost 25.

- `public int vrni(int[] indeksi)`

  Vrne vrednost elementa p-tabele `this`, določenega s podano tabelo indeksov. Lahko predpostavite, da je dolžina tabele `indeksi` enaka številu dimenzij p-tabele `this` in da so vsi indeksi veljavni.

- `public Ptabela podtabela(int[] indeksi)`

Vrne nov objekt tipa `Ptabela`, ki predstavlja kopijo pod-p-tabele p-tabele `this`, določene s podano tabelo indeksov. Lahko predpostavite, da je dolžina tabele `indeksi` manjša od števila dimenzij p-tabele `this` in da so vsi indeksi veljavni.

Na primer, če objekt `p` predstavlja p-tabelo velikosti $3 \times 4 \times 2$, potem bi klic

```
p.podtabela(new int[]{2})
```

vrnil kopijo tretje »strani« p-tabele `p`. Vrnjeni objekt bi torej predstavljal p-tabelo velikosti 4×2. Klic

```
p.podtabela(new int[]{2, 0})
```

pa bi vrnil kopijo prve vrstice tretje »strani« p-tabele `p`.

- `public String toString()`

  Vrne predstavitev p-tabele `this` v obliki niza. Enodimenzionalna p-tabela z d_1 elementi naj bo predstavljena z nizom $[a_0, a_1, \ldots, a_{d_1-1}]$, kjer so $a_0, a_1, \ldots, a_{d_1-1}$ elementi p-tabele. P-tabela velikosti $d_1 \times d_2 \times d_3 \times \ldots \times d_n$ naj bo predstavljena z nizom $[S_0, S_1, \ldots, S_{d_1-1}]$, kjer so $S_0, S_1, \ldots, S_{d_1-1}$ nizi, ki predstavljajo posamezne pod-p-tabele velikosti $d_2 \times d_3 \times \ldots \times d_n$.

Testni primer 8

Testni razred:

```
import java.util.Random;

public class Test08 {

    public static void main(String[] args) {
        // delali bomo s tabelo 3 × 4 × 2
        int a = 3, b = 4, c = 2;

        System.out.println("Inicializacija:");
        Ptabela p = new Ptabela(new int[]{a, b, c});
        System.out.println(p.toString());
        System.out.println("----------");

        System.out.println("Polnjenje z elementi od -99 do 99:");
        Random random = new Random(12345);
        for (int i = 0;  i < a;  i++) {
            for (int j = 0;  j < b;  j++) {
                for (int k = 0;  k < c;  k++) {
                    p.nastavi(new int[]{i, j, k}, random.nextInt(199) - 99);
                }
            }
        }
        System.out.println(p.toString());
        System.out.println("----------");

        System.out.println("Elementi in podtabele:");
        System.out.println("p[2][0][1] = " + p.vrni(new int[]{2, 0, 1}));
        System.out.println("p[2][1] = " + p.podtabela(new int[]{2, 1}).toString());
```

```
        System.out.println("p[1] = " + p.podtabela(new int[]{1}).toString());
        System.out.println("p = " + p.podtabela(new int[]{}).toString());
    }
}
```

Izhod:

```
Inicializacija:
[[[0, 0], [0, 0], [0, 0], [0, 0]],
 [[0, 0], [0, 0], [0, 0], [0, 0]],
 [[0, 0], [0, 0], [0, 0], [0, 0]]]
----------
Polnjenje z elementi od -99 do 99:
[[[-95, 17], [24, 57], [-43, -72], [-2, 43]],
 [[27, 13], [52, 69], [64, -31], [60, 73]],
 [[1, 61], [-8, 23], [-56, 36], [45, -71]]]
----------
Elementi in podtabele:
p[2][0][1] = 61
p[2][1] = [-8, 23]
p[1] = [[27, 13], [52, 69], [64, -31], [60, 73]]
p = [[[-95, 17], [24, 57], [-43, -72], [-2, 43]],
     [[27, 13], [52, 69], [64, -31], [60, 73]],
     [[1, 61], [-8, 23], [-56, 36], [45, -71]]]
```

Opomba: Zavoljo večje preglednosti smo izpis tridimenzionalnih p-tabel umetno prelomili. Metoda `toString` naj tovrstnih prelomov ne vstavlja.

Namig

Bi lahko p-tabelo simulirali s pomočjo enodimenzionalne tabele?

4.6 Štiri v vrsto

Naloga

Pri igri Štiri v vrsto igralca izmenično spuščata žetone (npr. prvi igralec rdeče, drugi pa zelene) v navpični pravokotni okvir z m vrsticami in n stolpci. Igralec na potezi spusti žeton v enega od stolpcev, ki še niso povsem polni.

Vrstice in stolpce igralnega okvirja označujemo z indeksi od 0 naprej. Indeks 0 predstavlja spodnjo (!) vrstico in levi stolpec. Prvi igralec (to je tisti, ki odigra prvo potezo v igri) ima indeks 0, drugi pa 1.

V vseh testnih razredih imajo parametri metod smiselne vrednosti. Na primer, parameter, ki podaja indeks vrstice, ima vrednost med 0 in vključno $m - 1$.

Napišite razred `StiriVVrsto`, ki omogoča simulacijo igre Štiri v vrsto. V razredu definirajte sledeče javno dostopne elemente:

- `public StiriVVrsto(int stVrstic, int stStolpcev)`

  Ustvari objekt, ki predstavlja začetno stanje igre Štiri v vrsto na okvirju s `stVrstic` (m) vrsticami in `stStolpcev` (n) stolpci. V vseh testnih primerih velja $m \in [1, 100]$ in $n \in [1, 100]$.

- `public int vrniSteviloVrstic()`

  Vrne število vrstic igralnega okvirja.

- `public int vrniSteviloStolpcev()`

  Vrne število stolpcev igralnega okvirja.

- `public boolean vrzi(int stolpec)`

  Če je stolpec z indeksom `stolpec` že poln, naj metoda ne naredi ničesar in naj zgolj vrne `false`, v nasprotnem primeru pa naj simulira potezo (met žetona v podani stolpec in predaja poteze nasprotniku) in vrne `true`.

- `public int naPotezi()`

  Vrne indeks igralca, ki je trenutno na potezi. Na začetku igre naj metoda torej vrne vrednost 0.

- `public int vsebina(int vrstica, int stolpec)`

  Če je polje v vrstici z indeksom `vrstica` in stolpcu z indeksom `stolpec` prazno, vrne vrednost -1, sicer pa vrne indeks igralca, ki mu pripada žeton v tem polju.

- `public int najdaljseZaporedje(int igralec)`

  Vrne dolžino najdaljšega strnjenega zaporedja žetonov igralca z indeksom `igralec` v neki vrstici, stolpcu ali diagonali.

- `public int izid()`

  Če je prvi igralec postavil najmanj štiri zaporedne žetone v neki vrstici, stolpcu ali diagonali, drugemu pa to ni uspelo, naj metoda vrne vrednost 0. Če se s tem dosežkom lahko ponaša drugi igralec, prvi pa ne, naj metoda vrne vrednost 1. Če sta najmanj štiri zaporedne žetone postavila oba ali pa nobeden od njiju, naj vrne vrednost -1.

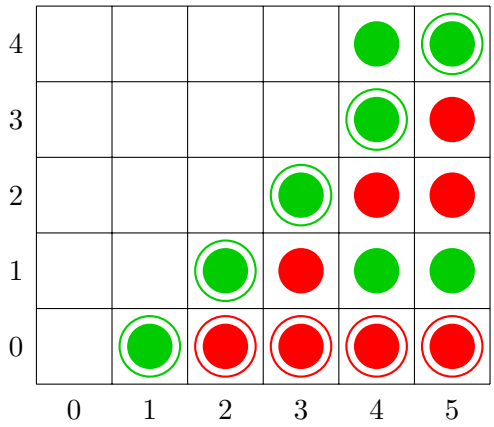
Testni primer 9

Testni razred (in pripadajoči izhod):

```
public class Test09 {

    public static void main(String[] args) {
        StiriVVrsto igra = new StiriVVrsto(5, 6);
        System.out.println(igra.naPotezi());   // 0
        System.out.println(igra.vrzi(2));      // true
        System.out.println(igra.vrzi(1));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(2));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(3));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(4));      // true
        System.out.println(igra.vrzi(5));      // true
        System.out.println(igra.vrzi(5));      // true
```

```
        System.out.println(igra.vrzi(5));       // true
        System.out.println(igra.vrzi(4));       // true
        System.out.println(igra.vrzi(4));       // false
        System.out.println(igra.vrzi(5));       // true
        System.out.println(igra.naPotezi());    // 1
        System.out.println(igra.vrzi(5));       // true
        System.out.println(igra.naPotezi());    // 0

        System.out.println(igra.vsebina(0, 0));   // -1
        System.out.println(igra.vsebina(2, 3));   // 1
        System.out.println(igra.vsebina(1, 3));   // 0
        System.out.println(igra.vsebina(3, 3));   // -1

        System.out.println(igra.najdaljseZaporedje(0));   // 4
        System.out.println(igra.najdaljseZaporedje(1));   // 5
        System.out.println(igra.izid());                  // -1
    }
}
```

Slika 4.2 prikazuje končno vsebino igralnega okvirja za gornji primer. Označeni sta najdaljši zaporedji žetonov za oba igralca. Ker sta oba igralca postavila najmanj štiri zaporedne žetone v neki vrstici, stolpcu ali diagonali, metoda `izid` vrne vrednost -1.



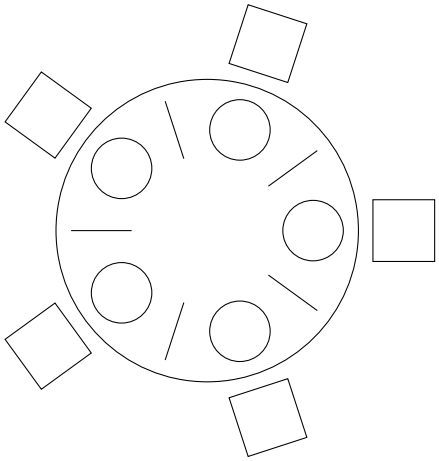
Slika 4.2: *Vsebina igralnega okvirja po koncu izvajanja kode v testnem razredu 9.*

4.7 Obedujoči filozofi

Naloga

Skupina filozofov sedi za okroglo mizo. Vsak filozof ima svoj krožnik s hrano, po dva soseda pa si delita jedilno paličico (slika 4.3). Filozofi večino časa razmišljajo, občasno pa kdo od njih postane lačen in želi pričeti jesti. Filozof za to opravilo potrebuje obe paličici, levo in desno. Paličici lahko vzame samo, če sta prosti — torej, če njegov levi sosed ne drži svoje desne paličice in če njegov desni sosed ne drži svoje leve paličice. Ko filozof prične jesti, prime obe paličici in ju drži, dokler ne konča. V tem času seveda preprečuje svojemu levemu in desnemu sosedu, da bi pričela s prehranjevanjem. Ko filozof preneha jesti, odloži (in s tem sprosti) obe paličici.[1]

[1] Gre za znan problem iz teorije operacijskih sistemov. Filozofi predstavljajo procese, ki tečejo v računalniku, paličice pa vire (npr. datoteke, naprave, procesorski čas, ...), ki jih procesi potrebujejo. Problem



Slika 4.3: *Miza za pet obedujočih filozofov.*

Za predstavitev posameznih filozofov napišite razred `Filozof` s sledečimi elementi:

- `public Filozof(String ime)`:

  Ustvari objekt, ki predstavlja filozofa za mizo. Parameter `ime` podaja njegovo ime. Filozof ob svoji »stvaritvi« ne drži niti leve niti desne paličice.

- `public String vrniIme()`:

  Vrne ime filozofa `this`.

- `public void nastaviSoseda(Filozof levi, Filozof desni)`:

  Nastavi oba soseda filozofa `this`. Levi sosed postane filozof `levi`, desni pa filozof `desni`.

- `public boolean jeLeviSosedOd(Filozof f)`:

  Vrne `true` natanko v primeru, če je filozof `this` levi sosed filozofa `f`.

- `public boolean jeDesniSosedOd(Filozof f)`:

  Vrne `true` natanko v primeru, če je filozof `this` desni sosed filozofa `f`.

- `public Filozof[] vrniSoseda()`:

  Vrne tabelo z dvema elementoma. Prvi element je referenca na levega, drugi pa referenca na desnega soseda filozofa `this`.

- `public int kolikoPalicicDrzi()`:

  Vrne število paličic, ki jih trenutno drži filozof `this`. To število je lahko samo 0, 1 ali 2.

- `public boolean primiLevo()`:

  Če filozof `this` že drži paličico na svoji levi, naj se ne zgodi nič, metoda pa naj vrne `true`. Če jo drži njegov levi sosed, naj se prav tako ne zgodi nič, metoda pa naj vrne `false`. Če je paličica prosta, naj jo prime (in s tem zasede) filozof `this`, metoda pa naj vrne `true`.

- `public boolean primiDesno()`:

obedujočih filozofov tako ilustrira borbo procesov za omejene vire.

Če filozof `this` že drži paličico na svoji desni, naj se ne zgodi nič, metoda pa naj vrne `true`. Če jo drži njegov desni sosed, naj se prav tako ne zgodi nič, metoda pa naj vrne `false`. Če je paličica prosta, naj jo prime (in s tem zasede) filozof `this`, metoda pa naj vrne `true`.

- `public void izpustiLevo()`:

  Če filozof `this` drži paličico na svoji levi, jo izpusti, sicer pa se ne zgodi nič.

- `public void izpustiDesno()`:

  Če filozof `this` drži paličico na svoji desni, jo izpusti, sicer pa se ne zgodi nič.

- `public static int steviloJedcev(Filozof[] filozofi)`:

  Vrne število filozofov v tabeli `filozofi`, ki držijo obe paličici.

- `public int steviloFilozofov()`:

  Vrne število filozofov za mizo, če privzamemo, da je filozof `this` eden od njih. Lahko predpostavite, da sta za mizo vsaj dva filozofa.

  Namig: pričnite s filozofom `this` in potujte po verigi desnih (ali levih) sosedov, dokler ne pridete spet do filozofa `this`.

Pri vseh metodah razen `vrniIme` in `nastaviSoseda` lahko predpostavite, da ima filozof `this` že pravilno (in konsistentno) nastavljena oba soseda. To pomeni, da vam ni treba preverjati, ali levi in desni sosed sploh obstajata.

Testni primer 9

Testni razred (in pripadajoči izhod):

```
public class Test09 {

    public static void main(String[] args) {
        Filozof slavoj = new Filozof("Slavoj");
        Filozof renata = new Filozof("Renata");
        Filozof mladen = new Filozof("Mladen");
        Filozof tine = new Filozof("Tine");
        Filozof spomenka = new Filozof("Spomenka");

        slavoj.nastaviSoseda(spomenka, renata);
        renata.nastaviSoseda(slavoj, mladen);
        mladen.nastaviSoseda(renata, tine);
        tine.nastaviSoseda(mladen, spomenka);
        spomenka.nastaviSoseda(tine, slavoj);

        System.out.println(renata.jeLeviSosedOd(slavoj));     // false
        System.out.println(spomenka.jeLeviSosedOd(slavoj));   // true
        System.out.println(tine.jeDesniSosedOd(mladen));      // true
        System.out.println(spomenka.jeDesniSosedOd(mladen));  // false
        System.out.println("---");

        izpisiSoseda(slavoj);      // Spomenka/Renata
        izpisiSoseda(renata);      // Slavoj/Mladen
        izpisiSoseda(mladen);      // Renata/Tine
```

```
        izpisiSoseda(tine);         // Mladen/Spomenka
        izpisiSoseda(spomenka);     // Tine/Slavoj
        System.out.println("---");

        System.out.println(slavoj.primiLevo());     // true
        System.out.println(renata.primiDesno());    // true
        System.out.println(mladen.primiLevo());     // false
        System.out.println(tine.primiDesno());      // true
        System.out.println(spomenka.primiDesno());  // false
        System.out.println(slavoj.primiLevo());     // true
        System.out.println(slavoj.primiDesno());    // true
        System.out.println(tine.primiLevo());       // true
        renata.izpustiLevo();
        slavoj.izpustiDesno();
        System.out.println(renata.primiLevo());     // true
        System.out.println("---");

        System.out.println(slavoj.kolikoPalicicDrzi());    // 1
        System.out.println(renata.kolikoPalicicDrzi());    // 2
        System.out.println(mladen.kolikoPalicicDrzi());    // 0
        System.out.println(tine.kolikoPalicicDrzi());      // 2
        System.out.println(spomenka.kolikoPalicicDrzi());  // 0
        System.out.println("---");

        Filozof[] filozofi = {slavoj, renata, mladen, tine, spomenka};
        System.out.println(Filozof.steviloJedcev(filozofi));  // 2
        System.out.println("---");

        System.out.println(slavoj.steviloFilozofov());    // 5
        System.out.println(renata.steviloFilozofov());    // 5
        System.out.println(mladen.steviloFilozofov());    // 5
        System.out.println(tine.steviloFilozofov());      // 5
        System.out.println(spomenka.steviloFilozofov());  // 5
    }

    private static void izpisiSoseda(Filozof filozof) {
        Filozof[] soseda = filozof.vrniSoseda();
        System.out.printf("%s/%s%n", soseda[0].vrniIme(), soseda[1].vrniIme());
    }
}
```

4.8 Krožki

Naloga

Učenci osnovne šole Jožeta Goriška obiskujejo različne krožke. Vsak krožek se izvaja ob fiksnem terminu (enkrat tedensko po dve uri), sprejme pa lahko največ k učencev (vrednost k je od krožka do krožka lahko različna). Učenec se lahko včlani v krožek natanko tedaj, ko so izpolnjeni vsi sledeči pogoji:

1. Učenec je v danem trenutku član manj kot m krožkov. Omejitev m je enaka za vse učence na šoli.

2. Krožek ima v danem trenutku manj kot k članov.
3. Termin krožka se ne prekriva z nobenim od terminov krožkov, ki jih učenec že obiskuje.

Napišite razreda `Ucenec` in `Krozek` tako, da bodo njuni objekti predstavljali posamezne učence oziroma krožke. Razred `Ucenec` naj vsebuje sledeče elemente (seveda lahko po potrebi dodajate tudi svoje):

- `public static void nastaviMaksObremenitev(int maksObremenitev)`

  Nastavi konstanto m (največje možno število krožkov, v katere je lahko včlanjen posamezen učenec) na vrednost `maksObremenitev`. V vsakem testnem primeru, v katerem se vsaj enkrat pokliče metoda `vclani`, se metoda `nastaviMaksObremenitev` pokliče natanko enkrat, in to takoj na začetku izvajanja programa.

- `public Ucenec(String ime, String priimek)`

  Ustvari objekt, ki predstavlja učenca s podanim imenom in priimkom. Učenec ob svoji »stvaritvi« ni član nobenega krožka.

- `public String vrniIP()`

  Vrne ime in priimek učenca `this` kot niz sledeče oblike:

  $ime␣priimek$

- `public boolean vclani(Krozek krozek)`

  Če je učenec `this` že član krožka `krozek`, naj metoda zgolj vrne `true`. Če učenec ne izpolnjuje pogojev za včlanitev v krožek, naj metoda zgolj vrne `false`. Če pa se učenec lahko včlani v krožek, naj metoda to stori in vrne `true`.

- `public void izclani(Krozek krozek)`

  Učenca `this` izpiše iz krožka `krozek`. Če učenec ni član podanega krožka, naj se ne zgodi nič.

- `public int steviloKrozkov()`

  Vrne število krožkov, v katere je včlanjen učenec `this`.

Razred `Krozek` pa naj vsebuje sledeče elemente:

- `public Krozek(String naziv, int dan, int ura, int kvota)`

  Ustvari objekt, ki predstavlja krožek s podanim nazivom. Krožek se izvaja v dnevu `dan` (1: ponedeljek, 2: torek, ...), prične pa se ob uri `ura`. Krožek lahko ima največ `kvota` (k) članov.

- `public String vrniNaziv()`:

  Vrne naziv krožka `this`.

- `public int steviloClanov()`:

  Vrne število članov krožka `this`.

Testni primer 8

Testni razred (in pripadajoči izhod):

```
public class Test08 {

    private static final String LOCILO = "--------------------";

    public static void main(String[] args) {
        Ucenec.nastaviMaksObremenitev(3);

        Ucenec anja = new Ucenec("Anja", "Antolinc");
        Ucenec bojan = new Ucenec("Bojan", "Bevk");
        Ucenec cvetka = new Ucenec("Cvetka", "Cirnik");
        Ucenec denis = new Ucenec("Denis", "Divjak");
        Ucenec eva = new Ucenec("Eva", "Erlah");

        Krozek matematika = new Krozek("matematika", 1, 13, 2);
        Krozek sah = new Krozek("sah", 1, 14, 3);
        Krozek dramatika = new Krozek("dramatika", 1, 16, 2);
        Krozek ples = new Krozek("ples", 3, 15, 4);
        Krozek nogomet = new Krozek("nogomet", 4, 13, 2);

        Ucenec[] ucenci = {anja, bojan, cvetka, denis, eva};
        Krozek[] krozki = {matematika, sah, dramatika, ples, nogomet};

        System.out.println(anja.vclani(matematika));   // true
        System.out.println(anja.vclani(sah));          // false
        System.out.println(anja.vclani(dramatika));    // true
        System.out.println(anja.vclani(ples));         // true
        System.out.println(anja.vclani(nogomet));      // false
        System.out.println(anja.vclani(matematika));   // true
        System.out.println(LOCILO);

        System.out.println(bojan.vclani(matematika));  // true
        System.out.println(bojan.vclani(dramatika));   // true
        System.out.println(bojan.vclani(nogomet));     // true
        System.out.println(LOCILO);

        System.out.println(cvetka.vclani(sah));        // true
        System.out.println(cvetka.vclani(dramatika));  // false
        System.out.println(cvetka.vclani(ples));       // true
        System.out.println(LOCILO);

        System.out.println(denis.vclani(sah));         // true
        System.out.println(denis.vclani(dramatika));   // false
        System.out.println(denis.vclani(ples));        // true
        System.out.println(LOCILO);

        System.out.println(eva.vclani(sah));           // true
        System.out.println(eva.vclani(dramatika));     // false
        System.out.println(eva.vclani(ples));          // true
        System.out.println(LOCILO);

        izpisiVse(ucenci, krozki);   // Anja Antolinc -> 3
```

```
                                    // Bojan Bevk -> 3
                                    // Cvetka Cirnik -> 2
                                    // Denis Divjak -> 2
                                    // Eva Erlah -> 2
                                    // matematika -> 2
                                    // sah -> 3
                                    // dramatika -> 2
                                    // ples -> 4
                                    // nogomet -> 1
        System.out.println(LOCILO);

        anja.izclani(matematika);
        bojan.izclani(nogomet);
        cvetka.izclani(ples);
        denis.izclani(sah);
        eva.izclani(nogomet);

        izpisiVse(ucenci, krozki);  // Anja Antolinc -> 2
                                    // Bojan Bevk -> 2
                                    // Cvetka Cirnik -> 1
                                    // Denis Divjak -> 1
                                    // Eva Erlah -> 2
                                    // matematika -> 1
                                    // sah -> 2
                                    // dramatika -> 2
                                    // ples -> 3
                                    // nogomet -> 0
        System.out.println(LOCILO);

        System.out.println(eva.vclani(dramatika));   // false
        System.out.println(eva.vclani(nogomet));     // true
        System.out.println(cvetka.vclani(nogomet));  // true
        System.out.println(bojan.vclani(nogomet));   // false
        System.out.println(anja.vclani(sah));        // true
        System.out.println(LOCILO);

        izpisiVse(ucenci, krozki);  // Anja Antolinc -> 3
                                    // Bojan Bevk -> 2
                                    // Cvetka Cirnik -> 2
                                    // Denis Divjak -> 1
                                    // Eva Erlah -> 3
                                    // matematika -> 1
                                    // sah -> 3
                                    // dramatika -> 2
                                    // ples -> 3
                                    // nogomet -> 2
    }

    private static void izpisiVse(Ucenec[] ucenci, Krozek[] krozki) {
        for (int i = 0;  i < ucenci.length;  i++) {
            System.out.printf("%s -> %d%n",
```

```
                                ucenci[i].vrniIP(), ucenci[i].steviloKrozkov());
        }
        for (int i = 0;  i < krozki.length;  i++) {
            System.out.printf("%s -> %d%n",
                                krozki[i].vrniNaziv(), krozki[i].steviloClanov());
        }
    }
}
```

4.9 Praštevila

Naloga

Napišite razred `Prastevila`, čigar objekt hrani t.i. *trenutno praštevilo*, ki ga je mogoče nastavljati, vračati in spreminjati. Razred naj vsebuje sledeče javno dostopne elemente:

- `public Prastevila()`

  Ustvari objekt, v katerem je trenutno praštevilo nastavljeno na 2.

- `public void nastaviTrenutno(int prastevilo)`

  Nastavi trenutno praštevilo na `prastevilo`. V vseh testnih primerih je `prastevilo` praštevilo z intervala $[2, 10^9]$.

- `public int vrniTrenutno()`

  Vrne trenutno praštevilo, ki ga hrani objekt `this`.

- `public int naslednje()`

  Trenutno praštevilo nastavi na najmanjše praštevilo, ki je večje od trenutnega, in dobljeno število tudi vrne.

- `public int prejsnje()`

  Če je trenutno praštevilo enako 2, naj metoda zgolj vrne 2, sicer pa naj trenutno praštevilo nastavi na največje praštevilo, ki je manjše od trenutnega, in dobljeno število tudi vrne.

Skupno število klicev metod `naslednje` in `prejsnje` je v vseh testnih primerih omejeno na 1000.

Testni primer 6

Testni razred (in pripadajoči izhod):

```
public class Test06 {

    public static void main(String[] args) {
        Prastevila prastevila = new Prastevila();
        System.out.println(prastevila.vrniTrenutno());  // 2
        System.out.println(prastevila.prejsnje());      // 2
        System.out.println(prastevila.vrniTrenutno());  // 2
        System.out.println(prastevila.naslednje());     // 3
        System.out.println(prastevila.vrniTrenutno());  // 3
```

```
        prastevila.nastaviTrenutno(17);
        System.out.println(prastevila.vrniTrenutno());  // 17
        System.out.println(prastevila.prejsnje());      // 13
        System.out.println(prastevila.prejsnje());      // 11
        System.out.println(prastevila.naslednje());     // 13
        System.out.println(prastevila.vrniTrenutno());  // 13

        prastevila.nastaviTrenutno(47);
        System.out.println(prastevila.vrniTrenutno());  // 47
        System.out.println(prastevila.naslednje());     // 53
        System.out.println(prastevila.naslednje());     // 59
        System.out.println(prastevila.prejsnje());      // 53
        System.out.println(prastevila.vrniTrenutno());  // 53
    }
}
```

4.10 Poti po grafu (⋆)

Naloga

Napišite program, ki izpiše vse aciklične poti med podanim začetnim in končnim vozliščem v podanem enostavnem usmerjenem grafu brez zank.

Vhod

V prvi vrstici vhoda je zapisano število vozlišč ($n \in [2, 100]$), v drugi pa število povezav ($m \in [1, n(n-1)]$). V vsaki od naslednjih m vrstic sta podani celi števili $u \in [0, n-1]$ in $v \in [0, n-1] \backslash \{u\}$, ločeni s presledkom, ki predstavljata indeksa začetnega in končnega vozlišča povezave. Sledita še dve vrstici, v katerih sta zapisani števili $v_z \in [0, n-1]$ in $v_k \in [0, n-1] \backslash \{v_z\}$, ki predstavljata indeksa začetnega in končnega vozlišča poti.

Izhod

Vsako pot izpišite v svoji vrstici. Pot izpišite tako, da po vrsti navedete indekse vozlišč na njej. Indeksi naj bodo ločeni s presledkom.

Poti naj se izpišejo v leksikografskem vrstnem redu. Pot $u_1 \to u_2 \to \ldots \to u_k$ naj se torej izpiše pred potjo $v_1 \to v_2 \to \ldots \to v_l$ natanko tedaj, ko je izpolnjen eden od sledečih pogojev:

- $k < l$ in $u_i = v_i$ za vse $i \in \{1, \ldots, k\}$;
- obstaja $i \in \{1, \ldots, k\}$, tako da velja $u_i < v_i$ in $u_j = v_j$ za vse $j < i$.

Število poti v nobenem testnem primeru ne bo večje od 100.

Testni primer 7

V tem testnem primeru iščemo poti od vozlišča 3 do vozlišča 1 v grafu na sliki 4.4.

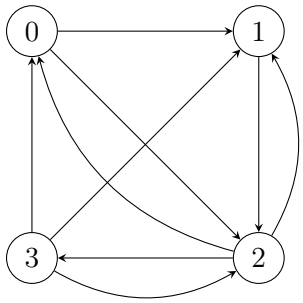
Vhod:

```
4
9
0 1
```

```
0 2
1 2
2 0
2 1
2 3
3 0
3 1
3 2
3
1
```

Izhod:

```
3 0 1
3 0 2 1
3 1
3 2 0 1
3 2 1
```



Slika 4.4: *Graf v testnem primeru 7.*

5 Dedovanje

5.1 Poštne pošiljke

Naloga

Na poštnem uradu razpošiljajo tri vrste pošiljk: navadna pisma, priporočena pisma in telegrame. Za vsako pošiljko sta podana naslovnik in vsebina. Za navadna pisma je poleg tega podana še razdalja do naslovnika, za priporočena pisma pa (poleg naslovnika, vsebine in razdalje) še pošiljatelj. Naslovniki, pošiljatelji in vsebine pošiljk so podani kot nizi, sestavljeni iz črk angleške abecede, števk in podčrtajev.

Cena oddaje navadnega pisma je odvisna zgolj od razdalje: za razdaljo od 0 do vključno $(R-1)$ dolžinskih enot znaša cena Z denarnih enot, za razdaljo od R do vključno $(2R-1)$ znaša $(Z+D)$ enot, za razdaljo od $2R$ do vključno $(3R-1)$ je cena enaka $(Z+2D)$ itd. Cena oddaje priporočenega pisma se izračuna tako, da se cena oddaje navadnega pisma pomnoži s faktorjem P. Cena telegrama je enaka cT, kjer je c število črk v vsebini telegrama (števke in podčrtaji ne štejejo).

Napišite program, ki prebere zaporedje poštnih pošiljk in ukaz (celo število med 1 in vključno 3), nato pa na podlagi ukaza proizvede ustrezni izpis:

- Če je ukaz enak 1, naj se izpišejo podatki o vseh pošiljkah. Za vsako pošiljko mora izpis vsebovati njeno vrsto, naslovnika, vsebino, morebitne dodatne podatke (odvisno od vrste pošiljke) in ceno.
- Če je ukaz enak 2, naj se izpišejo podatki o najdražji pošiljki (o prvi od njih, če jih je več).
- Če je ukaz enak 3, naj se za vsako priporočeno pismo izpišejo podatki o njegovi povratnici. Povratnica priporočenega pisma je priporočeno pismo, pri katerem je razdalja enaka kot pri izhodiščnem pismu, naslovnik in pošiljatelj sta med seboj zamenjana, vsebina pa je niz `povratnica`.

Vrstni red izpisa pošiljk na izhodu mora biti enak vrstnemu redu pošiljk na vhodu.

Vhod

V prvi vrstici vhoda je v tem vrstnem redu nanizanih pet celih števil z intervala $[1, 1000]$: Z, R, D, P in T. V drugi vrstici je podano celo število $n \in [1, 100]$. Sledi n vrstic s podatki o pošiljkah. V vsaki vrstici je najprej podana vrsta pošiljke (`navadnoPismo`, `priporocenoPismo` oziroma `telegram`), nato pa sledita naslovnik in vsebina. Pri navadnem

pismu sledi še razdalja, pri priporočenem pa razdalja in pošiljatelj. V zadnji vrstici vhoda je zapisano število $U \in \{1, 2, 3\}$, ki predstavlja ukaz.

Vsi nizi vsebujejo od 1 do 100 znakov. Vse razdalje so cela števila z intervala $[0, 1000]$. Podatki znotraj iste vrstice so med seboj ločeni s presledkom.

Izhod

Na izhodu izpišite podatke, ki jih zahteva ukaz. Zgledujte se po sledečih primerih. Zadnji podatek v vsaki vrstici je cena pošiljke.

Testni primer 1

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
1
```

Izhod:

```
T | Ana_Antolinc | vsebina_telegrama_1 | 144
PP | Branko_Bizjak | vsebina_pp_1 | 150 | Cvetka_Cevc | 180
NP | Danica_Dobravc | vsebina_np_1 | 75 | 40
T | Eva_Erker | zelo_dolga_VSEBINA_s_5t3v1lk4m1 | 198
NP | Franci_Furman | vsebina_np_2 | 49 | 30
NP | Gorazd_Gaber | vsebina_np_2 | 50 | 40
PP | Hinko_Hojc | vsebina_pp_2 | 50 | Iva_Intihar | 120
```

Testni primer 2

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
2
```

Izhod:

```
T | Eva_Erker | zelo_dolga_VSEBINA_s_5t3v1lk4m1 | 198
```

Testni primer 3

Vhod:

```
30 50 10 3 9
7
telegram Ana_Antolinc vsebina_telegrama_1
priporocenoPismo Branko_Bizjak vsebina_pp_1 150 Cvetka_Cevc
navadnoPismo Danica_Dobravc vsebina_np_1 75
telegram Eva_Erker zelo_dolga_VSEBINA_s_5t3v1lk4m1
navadnoPismo Franci_Furman vsebina_np_2 49
navadnoPismo Gorazd_Gaber vsebina_np_2 50
priporocenoPismo Hinko_Hojc vsebina_pp_2 50 Iva_Intihar
3
```

Izhod:

```
PP | Cvetka_Cevc | povratnica | 150 | Branko_Bizjak | 180
PP | Iva_Intihar | povratnica | 50 | Hinko_Hojc | 120
```

Napotki

Nize beremo s klicem `sc.next()`, pri čemer je `sc` objekt razreda `Scanner`. Če je `s` niz (objekt tipa `String`), potem njegovo dolžino (število znakov) pridobimo s klicem `s.length()`, znak z indeksom `i` pa s klicem `s.charAt(i)`. Na vprašanje, ali je podani znak `c` (spremenljivka tipa `char`) črka, odgovorimo s klicem `Character.isLetter(c)`.

5.2 Geometrijska telesa

Naloga

Na vhodu so zapisani osnovni podatki o različnih geometrijskih telesih: kvadrih, kockah in kroglah. Kvader je določen s tremi stranicami, kocka z eno samo, krogla pa s polmerom. Napišite program, ki prebere podatke o telesih ter izpiše njihove prostornine in površine, pri kvadrih in kockah pa tudi mreže. Prostornine in površine naj bodo zaokrožene na najbližje celo število.

Pri risanju mreže naj se zgornja in spodnja stranica kvadra oz. kocke prikažeta z znaki `o`, sprednja in zadnja z znaki `*`, leva in desna pa z znaki `+`. Znaki v isti vrstici naj bodo med seboj ločeni s presledkom. Na primer, mreža kvadra s stranicami $a = 2$, $b = 3$ in $c = 4$ naj se nariše tako:

```
        * * *
        * * *
        * * *
        * * *
+ + + + o o o + + + +
+ + + + o o o + + + +
        * * *
        * * *
        * * *
        * * *
        o o o
        o o o
```

oziroma (s prikazanimi presledki)

```
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
+␣+␣+␣+␣o␣o␣o␣+␣+␣+␣+
+␣+␣+␣+␣o␣o␣o␣+␣+␣+␣+
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣*␣*␣*
␣␣␣␣␣␣␣␣o␣o␣o
␣␣␣␣␣␣␣␣o␣o␣o
```

V izpisu morajo biti telesa urejena po padajočih prostorninah. Urejanje mora biti stabilno: telesa z enakimi prostorninami morajo biti na izhodu izpisana v istem medsebojnem vrstnem redu, kot so podana na vhodu.

Vhod

V prvi vrstici vhoda je podano celo število $n \in [1, 100]$, nato pa sledi n vrstic s podatki o posameznih telesih. Na začetku vsake vrstice je zapisano število 1 (to število predstavlja kvader), 2 (kocka) oziroma 3 (krogla), nato pa sledijo dolžine stranic (pri kvadru), dolžina stranice (pri kocki) oziroma polmer (pri krogli). Vsi navedeni podatki so cela števila z intervala $[1, 100]$, med seboj pa so ločeni s presledkom.

Izhod

Pri izpisu na izhod se zgledujte po primeru v nadaljevanju. Ne izpisujte odvečnih presledkov ali praznih vrstic!

Testni primer 1

Vhod:

```
6
2 4
1 4 2 8
3 4
2 3
3 2
1 8 4 2
```

Izhod:

```
krogla
r = 4
V = 268
P = 201
======
kocka
a = 4
V = 64
```

```
P = 96
        * * * *
        * * * *
        * * * *
        * * * *
+ + + + o o o o + + + +
+ + + + o o o o + + + +
+ + + + o o o o + + + +
+ + + + o o o o + + + +
        * * * *
        * * * *
        * * * *
        * * * *
        o o o o
        o o o o
        o o o o
        o o o o
======
kvader
a = 4
b = 2
c = 8
V = 64
P = 112
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                * *
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
+ + + + + + + + o o + + + + + + + + +
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                * *
                o o
                o o
                o o
                o o
======
kvader
a = 8
```

```
b = 4
c = 2
V = 64
P = 112
    * * * *
    * * * *
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
+ + o o o o + +
    * * * *
    * * * *
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
    o o o o
======
krogla
r = 2
V = 34
P = 50
======
kocka
a = 3
V = 27
P = 54
      * * *
      * * *
      * * *
+ + + o o o + + +
+ + + o o o + + +
+ + + o o o + + +
      * * *
      * * *
      * * *
      o o o
      o o o
      o o o
======
```

5.3 Seznam (⋆)

Naloga

Seznam je podatkovna struktura, ki predstavlja zaporedje elementov. Obstajata dve vrsti seznamov:

Prazen seznam: To je seznam, ki ne vsebuje nobenih elementov.

Neprazen seznam: To je seznam, ki je sestavljen iz *glave* in *repa*. Glava je prvi element seznama (v tej nalogi bo to vedno neko celo število), rep pa je seznam (prazen ali neprazen). Seznam z elementi 3, 5 in 7 je sestavljen iz glave 3 in repa, ki je seznam, sestavljen iz glave 5 in repa, ki je seznam, sestavljen iz glave 7 in praznega seznama. Če prazen seznam zapišemo kot `[]`, neprazen pa kot `[`*glava* `|` *rep*`]`, bi lahko seznam z elementi 3, 5 in 7 zapisali kot `[3 | [5 | [7 | []]]]`.

To definicijo lahko prevedemo v hierarhijo razredov, ki jo sestavljajo abstrakten razred `Seznam` ter njegova podrazreda `Prazen` in `Neprazen`. Realizirajte vse tri razrede. Razred `Prazen` mora vsebovati konstruktor

```
public Prazen()
```

ki izdela objekt, ki predstavlja prazen seznam. Razred `Neprazen` pa mora vsebovati konstruktor

```
public Neprazen(int glava, Seznam rep)
```

ki izdela objekt, ki predstavlja seznam s podano glavo in repom. Na primer, objekt, ki predstavlja seznam z elementi 3, 5 in 7, bi s pomočjo navedenih konstruktorjev ustvarili takole:

```
Seznam s = new Neprazen(3, new Neprazen(5, new Neprazen(7, new Prazen())));
```

V nadaljevanju bomo podali množico javno dostopnih metod, ki jih mora ponujati razred `Seznam`. Premislite, katere od teh metod naj bodo v razredu `Seznam` abstraktne (in definirane v obeh podrazredih), katere pa je možno definirati že v razredu `Seznam`. Nobena metoda ne spremeni seznama `this` in nobena ni daljša od nekaj vrstic.

Razred `Seznam` mora ponujati sledeče metode:

- `public int glava()`:

  Če je seznam `this` neprazen, vrne njegovo glavo, sicer pa sproži izjemo tipa `NoSuchElementException`:

  ```
  throw new java.util.NoSuchElementException();
  ```

- `public Seznam rep()`:

  Če je seznam `this` neprazen, vrne njegov rep, sicer pa sproži izjemo tipa `NoSuchElementException`:

- `public boolean jePrazen()`:

  Vrne `true` natanko v primeru, če je seznam prazen.

- `public Seznam dodajZ(int element)`:

  Vrne nov seznam, ki ima v glavi podani element, njegov rep pa je seznam `this`.

- `public Seznam dodajK(int element)`:

Vrne nov seznam, ki je enak seznamu `this`, le da ima na koncu dodan še podani element.

Pri praznem seznamu ta metoda deluje enako kot metoda `dodajZ`, pri nepraznem pa metoda `dodajK` izdela in vrne nov neprazen seznam, čigar glava je enaka glavi seznama `this`, rep pa je enak seznamu, ki ga dobimo, če na konec repa seznama `this` dodamo podani element.

- `public Seznam dodajU(int element)`:

  Ob predpostavki, da je seznam `this` naraščajoče urejen, ta metoda vrne nov seznam, ki je enak seznamu `this`, le da ima podani element vstavljen na mestu, kamor spada po velikosti. Na primer, če seznam `s` po vrsti vsebuje elemente 3, 5 in 7, potem klic `s.dodajU(6)` vrne seznam [3, 5, 6, 7], klic `s.dodajU(2)` pa seznam [2, 3, 5, 7].

  Pri nepraznem seznamu upoštevajte dve možnosti: (1) podani element je manjši ali enak glavi seznama `this`; (2) podani element je večji od glave.

- `public boolean vsebuje(int element)`:

  Vrne `true` natanko v primeru, če seznam vsebuje podani element. Prazen seznam elementa gotovo ne vsebuje, pri nepraznem seznamu pa moramo najprej preveriti glavo, če ta ni enaka iskanemu elementu, pa preiščemo še rep.

- `public Seznam odstrani(int element)`:

  Vrne nov seznam, ki je enak seznamu `this`, le da je v njem odstranjen prvi element, ki je enak podanemu elementu. Na primer, če je seznam `s` enak [7, 3, 6, 3, 2, 3], potem klic `s.odstrani(3)` vrne seznam [7, 6, 3, 2, 3]. Če seznam `this` podanega elementa ne vsebuje, naj bo rezultat metode enak seznamu `this`.

- `public Seznam uredi()`:

  Vrne naraščajoče urejeno kopijo seznama `this`. Na primer, klic metode nad seznamom [7, 6, 3, 2, 3] bi vrnil seznam [2, 3, 3, 6, 7].

  Namig: pri nepraznem seznamu najprej uredimo rep.

- `public Seznam odstraniDuplikate()`:

  Vrne nov seznam, ki je enak seznamu `this`, le da v njem vsak element nastopa samo po enkrat. Ohraniti se mora le *zadnja* pojavitev elementa. Na primer, klic metode nad seznamom [7, 3, 6, 3, 2, 3, 7, 2, 9] bi vrnil seznam [6, 3, 7, 2, 9].

- `public String toString()`:

  Vrne niz oblike

  `[`a_1`,␣`a_2`,␣...,␣`a_n`]`

  kjer so a_1, a_2, ..., a_n elementi seznama `this`. Na primer, klic metode nad seznamom [7, 6, 3, 2, 3] bi vrnil sledeči niz:

  ```
  [7, 6, 3, 2, 3]
  ```

  Pri realizaciji metode `toString` vam bo morda koristila pomožna metoda, ki izdela niz brez oklepajev.

Testni primer 10

Testni razred (in pripadajoči izhod):

```
public class Test10 {

    public static void main(String[] args) {
        Seznam seznam = new Prazen();
        System.out.println(seznam.jePrazen());  // true
        System.out.println(seznam.toString());  // []

        seznam = seznam.dodajU(30);
        seznam = seznam.dodajU(10);
        seznam = seznam.dodajU(40);
        seznam = seznam.dodajU(20);
        seznam = seznam.dodajU(60);
        seznam = seznam.dodajU(50);
        seznam = seznam.dodajU(10);
        seznam = seznam.dodajU(20);
        System.out.println(seznam.jePrazen());  // false
        System.out.println(seznam.toString());
            // [10, 10, 20, 20, 30, 40, 50, 60]

        seznam = seznam.dodajZ(50);
        seznam = seznam.dodajZ(60);
        seznam = seznam.dodajK(20);
        seznam = seznam.dodajK(50);
        System.out.println(seznam.toString());
            // [60, 50, 10, 10, 20, 20, 30, 40, 50, 60, 20, 50]

        seznam = seznam.odstrani(30);
        seznam = seznam.odstrani(50);
        seznam = seznam.odstrani(30);
        System.out.println(seznam.toString());
            // [60, 10, 10, 20, 20, 40, 50, 60, 20, 50]

        Seznam urejen = seznam.uredi();
        System.out.println(urejen.toString());
            // [10, 10, 20, 20, 20, 40, 50, 50, 60, 60]

        seznam = seznam.odstraniDuplikate();
        System.out.println(seznam.toString());  // [10, 40, 60, 20, 50]
    }
}
```

5.4 Naravno število (⋆)

Naloga

Cilj te naloge je predstaviti naravna števila (1, 2, 3, . . .) in realizirati nekatere operacije nad njimi s pomočjo dedovanja in rekurzije. Ker bi lahko nalogo brez posebnih težav razširili na celotno množico celih števil, se izkaže, da celoštevilskih podatkovnih tipov pravzaprav sploh ne potrebujemo. Ta drzna trditev pa velja le na konceptualni ravni, saj časovna in prostorska učinkovitost operacij, ki jih boste realizirali v tej nalogi, ne bo ravno najboljša. Vendar pa naj tokrat estetske vrednote prevladajo nad materialnimi!

Množico naravnih števil lahko definiramo na sledeči način (Peanovi aksiomi):

- Število 1 je naravno število.
- Naslednik naravnega števila je tudi naravno število.

To definicijo lahko prevedemo v hierarhijo razredov, ki jo sestavljajo abstrakten razred `NaravnoStevilo` ter njegova podrazreda `Ena` in `Naslednik`. Realizirajte vse tri razrede. Razred `Ena` mora vsebovati konstruktor

```
public Ena()
```

ki izdela objekt, ki predstavlja naravno število 1. Razred `Naslednik` pa naj vsebuje konstruktor

```
public Naslednik(NaravnoStevilo stevilo)
```

ki izdela objekt, ki predstavlja naslednika podanega naravnega števila. Na primer, objekt, ki predstavlja naravno število 3, bi s pomočjo navedenih konstruktorjev ustvarili takole:

```
NaravnoStevilo tri = new Naslednik(new Naslednik(new Ena()));
```

V nadaljevanju bomo podali množico javno dostopnih metod, ki jih mora ponujati razred `NaravnoStevilo`. Premislite, katere od teh metod naj bodo v razredu `NaravnoStevilo` abstraktne (in definirane v obeh podrazredih), katere pa je možno definirati že v razredu `NaravnoStevilo`. Nobena metoda ne spremeni naravnega števila `this` in nobena ni daljša od nekaj vrstic.

Razred `NaravnoStevilo` naj ponuja sledeče metode:

- `public boolean jeEna()`:

  Vrne `true` natanko v primeru, če objekt `this` predstavlja število 1.

- `public NaravnoStevilo naslednik()`:

  Vrne naravno število, ki je naslednik naravnega števila `this`.

- `public NaravnoStevilo predhodnik()`:

  Vrne naravno število, ki je predhodnik naravnega števila `this`. Upoštevajte, da je predhodnik števila `new Naslednik(`n`)` kar število n. Ker število 1 nima predhodnika, naj metoda zanj sproži izjemo tipa `NoSuchElementException`:

  ```
  throw new java.util.NoSuchElementException();
  ```

- `public NaravnoStevilo vsota(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je vsota naravnega števila `this` in števila `stevilo`. Namig: primer $1 + n$ sodi v razred `Ena`, primer $m + n$ za $m > 1$ pa v razred `Naslednik`.

- `public NaravnoStevilo razlika(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je razlika naravnega števila `this` in števila `stevilo`. Če rezultat razlike ni naravno število, naj metoda sproži izjemo tipa `NoSuchElement-Exception`.

- `public NaravnoStevilo zmnozek(NaravnoStevilo stevilo)`:

  Vrne naravno število, ki je zmnožek naravnega števila `this` in števila `stevilo`.

- `public String toString()`:

Če objekt `this` predstavlja število 1, vrne niz `1`, sicer pa vrne niz `s(`t`)`, kjer je t niz, ki ga metoda `toString` vrne za predhodnika števila `this`. Na primer, za število 3 naj metoda vrne niz `s(s(1))`.

- `public int toInt()`:

  Vrne celo število, ki ga predstavlja objekt `this`.

- `public static NaravnoStevilo ustvariIzInt(int n)`:

  Ustvari objekt, ki predstavlja podano pozitivno celo število `n`. To metodo morate seveda definirati v razredu `NaravnoStevilo`.

Testni primer 10

Testni razred (in pripadajoči izhod):

```
public class Test10 {

    public static void main(String[] args) {
        NaravnoStevilo ena = new Ena();
        NaravnoStevilo dve = ena.naslednik().naslednik().predhodnik();
        NaravnoStevilo pet = new Naslednik(new Naslednik(new Naslednik(dve)));
        NaravnoStevilo osem = NaravnoStevilo.ustvariIzInt(10).
                                  predhodnik().predhodnik();

        System.out.println(ena.jeEna());  // true
        System.out.println(dve.jeEna());  // false
        System.out.println(dve.predhodnik().jeEna());  // true

        NaravnoStevilo a = dve.vsota(pet);
        NaravnoStevilo b = osem.razlika(dve);
        NaravnoStevilo c = pet.zmnozek(dve);
        NaravnoStevilo d = pet.razlika(dve).razlika(dve);
        NaravnoStevilo e = ena.vsota(pet).zmnozek(osem.razlika(ena));

        System.out.println(a.toInt());     // 7
        System.out.println(b.toString());  // s(s(s(s(s(1)))))
        System.out.println(c.toInt());     // 10
        System.out.println(d.toString());  // 1
        System.out.println(e.toInt());     // 42
    }
}
```

6 Vsebovalniki

6.1 Zrcalna slika seznama

Napišite razred `ZrcalnaSlikaSeznama` z metodo

```
public static <T> List<T> zrcalnaSlika(List<T> seznam)
```

ki vrne nov seznam, ki vsebuje elemente podanega seznama v obratnem vrstnem redu. Na primer, za seznam [30, 20, 50, 40] naj metoda vrne seznam [40, 50, 20, 30].

6.2 Sploščitev seznama

Napišite razred `SploscitevSeznama` z metodo

```
public static <T> List<T> splosci(List<List<T>> seznami)
```

ki izdela in vrne nov seznam, v katerem so po vrsti nanizani elementi notranjih seznamov podanega seznama. Na primer, za seznam [[30], [20, 50, 10], [30, 40]] naj metoda vrne seznam [30, 20, 50, 10, 30, 40].

6.3 Presek množic

Napišite razred `PresekMnozic` z metodo

```
public static <T> Set<T> presek(Collection<Set<T>> mnozice)
```

ki vrne presek množic iz podane zbirke. Na primer, za zbirko ⟨{2, 4, 5, 7, 8, 9}, {4, 6, 7, 8, 9, 10}, {1, 2, 4, 5, 7, 8}⟩ naj metoda vrne množico {4, 7, 8}.

6.4 Manjkajoče besede

Naloga

Napišite program, ki prebere dve množici besed in v leksikografskem vrstnem redu izpiše vse besede, ki nastopajo v prvi množici, ne pa tudi v drugi. Vsako besedo naj izpiše samo po enkrat.

Vhod

Vhod je sestavljen iz štirih vrstic. Prva podaja število besed v prvi množici ($m \in [1, 1000]$), druga vsebuje m besed, ki pripadajo prvi množici, tretja podaja število besed v drugi množici ($n \in [1, 1000]$), četrta pa vsebuje n besed, ki pripadajo drugi množici. V vrsticah, ki podajata množici besed, so besede med seboj ločene s presledkom. Besede so sestavljene zgolj iz velikih in malih črk angleške abecede, števk in podčrtajev. Nobena beseda ni daljša od 100 znakov.

Izhod

Izpišite iskane besede. Vsaka beseda naj bo izpisana v svoji vrstici.

Testni primer 1

Vhod:

```
8
medtem ko je programski jezik java objektno usmerjen
6
je programski jezik C proceduralno usmerjen
```

Izhod:

```
java
ko
medtem
objektno
```

6.5 Potenčna množica

Napišite razred `PotencnaMnozica` z metodo

```
public static <T> Set<Set<T>> potencna(Set<T> mnozica)
```

ki vrne množico vseh podmnožic podane množice. Na primer, za množico {1, 2, 3} naj metoda vrne množico {{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}.

6.6 Pogostost besed

Naloga

Napišite program, ki prebere zaporedje besed in izpiše seznam besed in njihovih pogostosti. Seznam naj bo urejen po padajoči pogostosti, enako pogoste besede pa naj se izpišejo v leksikografskem vrstnem redu.

Vhod

Vhod je sestavljen iz ene same vrstice, ta pa vsebuje besede, ločene s presledkom. Besede so sestavljene zgolj iz velikih in malih črk angleške abecede, števk in podčrtajev.

Izhod

Za vsako različno besedo v vhodnem besedilu izpišite vrstico sledeče oblike:

beseda `(`*pogostost*`)`

Testni primer 1

Vhod:

```
marko skace marko skace po zeleni trati aj aj aj aj aj po zeleni trati
```

Izhod:

```
aj (5)
marko (2)
po (2)
skace (2)
trati (2)
zeleni (2)
```

6.7 Obrat slovarja

Napišite razred `ObratSlovarja` z metodo

```
public static<K, V> Map<V, Set<K>> obrni(Map<K, V> slovar)
```

ki vrne nov slovar, ki objekt v preslika v množico vseh objektov k, ki jih podani slovar preslika v objekt v. Na primer, za slovar s preslikavami

Slovenija $\mapsto$ 4,
Avstrija $\mapsto$ 8,
Italija $\mapsto$ 6,
Hrvaška $\mapsto$ 5,
Češka $\mapsto$ 4,
Slovaška $\mapsto$ 5,
Srbija $\mapsto$ 8,
Belgija $\mapsto$ 4

naj metoda vrne slovar s preslikavami

4 $\mapsto$ {Slovenija, Češka, Belgija},
5 $\mapsto$ {Hrvaška, Slovaška},
6 $\mapsto$ {Italija},
8 $\mapsto$ {Avstrija, Srbija}.

6.8 Slovar dvobojev

Napišite razred `SlovarDvobojev` z metodo

```
public static NavigableMap<String, NavigableMap<String, Integer>>
        partije2slovar(List<Partija> partije)
```

ki na podlagi seznama šahovskih partij izdela in vrne slovar, ki igralca a preslika v slovar, ki igralca $b \neq a$ preslika v skupno število točk, ki jih je igralec a zbral v dvobojih z igralcem b. Igralci so enolično določeni z imeni. Zmaga prinese 2 točki, remi 1 točko, poraz pa 0 točk.[1] Slovar naj vsebuje vnos za vsak par igralcev a in b, kjer je $a \neq b$. Če igralca a in b nista odigrala nobene partije, naj izraz `s.get(`a`).get(`b`)` potemtakem ne sproži izjeme, ampak naj vrne 0. Pri vseh slovarjih (pri zunanjem in notranjih) naj bodo ključi leksikografsko urejeni.

[1] V šahu zmaga dejansko prinese eno točko, remi pa pol točke, vendar pa se želimo izogniti decimalkam.

Razred `Partija` je definiran takole:

```
public class Partija {
    private String beli;    // ime igralca z belimi figurami
    private String crni;    // ime igralca s črnimi figurami
    private int izid;       // izid z vidika belega (2: zmaga, 1: remi, 0: poraz)

    public Partija(String beli, String crni, int izid) {
        this.beli = beli;
        this.crni = crni;
        this.izid = izid;
    }
}
```

Na primer, če je Ana premagala Bojana, Bojan Cvetko, Cvetka pa je remizirala z Ano, naj metoda vrne slovar s sledečimi preslikavami:

Ana $\mapsto$ {Bojan $\mapsto$ 2, Cvetka $\mapsto$ 1},
Bojan $\mapsto$ {Ana $\mapsto$ 0, Cvetka $\mapsto$ 2},
Cvetka $\mapsto$ {Ana $\mapsto$ 1, Bojan $\mapsto$ 0}.

7 Vmesniki in lambde

7.1 Zadnje ujemanje

Napišite razred `ZadnjeUjemanje` z metodo

```
public static <T> T zadnji(Collection<T> zbirka, Predicate<T> pogoj)
```

ki vrne zadnji element zbirke, ki izpolnjuje podani pogoj, oziroma `null`, če noben element zbirke ne izpolnjuje pogoja. Lahko predpostavite, da je vrstni red elementov v zbirki natančno določen (ta lastnost med drugim velja za zbirke tipov `List` in `NavigableSet`).

7.2 Kumulativa

Napišite razred `Kumulativa` z metodo

```
public static <T> List<T> kumulativa(List<T> seznam, BinaryOperator<T> operator)
```

ki vrne kumulativo podanega seznama glede na podani dvojiški operator (označimo ga s $\circ$). Če podani seznam vsebuje elemente s_0, s_1, $\ldots$, s_{n-1}, potem kumulativo tvorijo elementi t_0, t_1, $\ldots$, t_{n-1}, pri čemer je $t_0 = s_0$ in $t_i = t_{i-1} \circ s_i$ za vsak $i \in \{1, \ldots, n-1\}$. Na primer, kumulativa seznama [3, 7, −2, 9, 6] glede na operacijo seštevanja je seznam [3, 10, 8, 17, 23].

Metodo `kumulativa` nato uporabite v metodi

```
public static <T> List<T> doslejNajvecji(List<T> seznam, Comparator<T> primerjalnik)
```

ki za podani seznam vrne nov seznam iste dolžine, v katerem je i-ti element enak največjemu (glede na podani primerjalnik) izmed prvih i elementov podanega seznama. Na primer, pri seznamu [3, 7, −2, 9, 6] in običajni urejenosti števil bi dobili rezultat [3, 7, 7, 9, 9].

7.3 Urejanje po vrednostih funkcije

Napišite razred `UrejanjePoFunkciji` z metodo

```
public static <T, R extends Comparable<R>> void urediPoFunkciji(
        List<T> seznam, Function<T, R> funkcija)
```

ki podani seznam uredi glede na vrednosti podane funkcije na elementih seznama. Če za funkcijo f in elementa seznama p in q velja $f(p) < f(q)$, potem se mora v urejenem

seznamu element p nahajati pred elementom q. V metodi si pomagajte s klicem metode `sort` iz razreda `List`.

Na primer, če seznam $[-3, 5, -7, 8, 2, -6]$ uredimo glede na funkcijo $f(x) = |x|$, dobimo seznam $[2, -3, 5, -6, -7, 8]$, urejanje glede na funkcijo $f(x) = -x$ pa nam dá seznam $[8, 5, 2, -3, -6, -7]$.

Zapis `R extends Comparable<R>` v seznamu tipskih parametrov pove, da mora biti tip `R` podtip tipa `Comparable<R>` (če je `R` razred, mora implementirati vmesnik `Comparable<R>`). Generični tip `R` lahko potemtakem nadomestimo s tipom `Integer`, `String` ali `Cas`, ne pa s tipom `Object` ali `Oseba`.

Metodo `urediPoFunkciji` nato uporabite še v sledečih metodah:

- `public static void urediPoAbsolutniVrednosti(List<Integer> stevila)`

  Podani seznam uredi po absolutnih vrednostih posameznih elementov.

- `public static void urediPoDolzini(List<String> nizi)`

  Podani seznam uredi po dolžinah posameznih nizov.

- `public static <T extends Comparable<T>> List<Integer> vrstniRed(List<T> seznam)`

  Vrne nov seznam, v katerem i-ti element vsebuje indeks elementa podanega seznama, ki bi se v naravno urejeni različici podanega seznama nahajal na i-tem mestu. Na primer, za seznam [bojan, darja, eva, ana, cvetko] naj metoda vrne seznam [3, 0, 4, 1, 2].

7.4 Obrat primerjalnika

Napišite razred `ObratPrimerjalnika` z metodo

```
public static <T> Comparator<T> obrni(Comparator<T> primerjalnik)
```

ki vrne primerjalnik, glede na katerega sodi objekt a pred objekt b natanko tedaj, ko objekt b glede na podani primerjalnik sodi pred objekt a.

Metodo `obrni` nato uporabite v metodi

```
public static <T> void urediPadajoce(List<T> seznam, Comparator<T> primerjalnik)
```

ki podani seznam padajoče uredi glede na podani primerjalnik. Pomagajte si tudi z metodo `sort` iz razreda `List`.

7.5 Comparable → Comparator

Napišite razred `ComparableVComparator` z metodo

```
public static <T extends Comparable<T>> Comparator<T> pretvori()
```

ki vrne primerjalnik, ki objekta primerja glede na njuno naravno urejenost. Zapis `T extends Comparable<T>` v seznamu tipskih parametrov pove, da mora biti tip `T` podtip tipa `Comparable<T>` (če je `T` razred, mora implementirati vmesnik `Comparable<T>`).

7.6 Kombinacija primerjalnikov

Napišite razred `KombinacijaPrimerjalnikov` z metodo

`public static <T> Comparator<T> kombinacija(List<Comparator<T>> primerjalniki)`

ki vrne primerjalnik, ki par elementov tipa `T` primerja s prvim primerjalnikom v podanem nepraznem seznamu in vrne rezultat primerjave, če se elementa po tem kriteriju razlikujeta, sicer pa elementa primerja še z drugim primerjalnikom v podanem seznamu. Če se tokrat razlikujeta, vrne rezultat primerjave, sicer pa ju primerja še s tretjim primerjalnikom itd. Če se elementa po nobenem primerjalniku ne razlikujeta, je končni rezultat metode enak 0.

7.7 Iterator po tabeli

Razred `OvojnikTabele` ...

```
public class OvojnikTabele {
    private Object[] tabela;

    public OvojnikTabele(Object[] tabela) {
        this.tabela = tabela;
    }
}
```

... dopolnite tako, da se bo zanka v sledeči kodi sprehodila po elementih tabele `t`:

```
Object[] t = ...;
OvojnikTabele tab = new OvojnikTabele(t);
for (Object element: tab) {
    ...
}
```

7.8 Sprehod s preskoki

Napišite razred `SprehodSPreskoki<T>`, ki omogoča sprehajanje (z zanko for-each) po podanem seznamu s podanim korakom. Razred naj vsebuje konstruktor

`public SprehodSPreskoki(List<T> seznam, int korak)`

pri čemer parameter `seznam` predstavlja seznam, po katerem se bomo sprehajali, neničelni parameter `korak` pa podaja korak sprehoda (k). Če je korak k pozitiven, potem sprehod po seznamu z zanko for-each pričnemo z elementom na indeksu 0, nato pa po vrsti (dokler ne prispemo do konca seznama) obiščemo elemente na indeksih k, $2k$, $3k$ itd. Če je korak k negativen, pa sprehod pričnemo na zadnjem elementu seznama, nato pa indeks po vrsti zmanjšujemo za $|k|$, dokler ne prispemo do začetka seznama. Na primer, koda

```
List<Integer> seznam = List.of(20, 21, 22, 23, 24, 25, 26, 27, 28, 29);
SprehodSPreskoki naprej = new SprehodSPreskoki(seznam, 4);
for (int element: naprej) {
    System.out.print(element + " ");
}
System.out.println();
SprehodSPreskoki nazaj = new SprehodSPreskoki(seznam, -3);
for (int element: nazaj) {
    System.out.print(element + " ");
```

```
}
System.out.println();
```

bi izpisala

```
20 24 28
29 26 23 20
```

7.9 Kombinacija iteratorjev

Iterator ima k *korakov*, če potrebuje k klicev metode `next`, da prispe do konca svojega sprehoda. *Kombinacija* iteratorjev I_0, I_1, ..., I_{n-1}, od katerih imajo vsi po k korakov, je iterator I z nk koraki, pri katerem i-ti klic metode `next` (za $i \in \{0, \ldots, nk-1\}$) vrne trenutni element pod iteratorjem $I_{i \bmod n}$ in premakne ta iterator na naslednji element.

Napišite razred `KombinacijaIteratorjev<T>` z metodo

```
public static <T> Iterator<T> kombinacija(List<Iterator<T>> iteratorji)
```

ki vrne kombinacijo iteratorjev iz podanega seznama, pri čemer lahko predpostavite, da imajo vsi iteratorji enako število korakov.