

Dodatne naloge
pri predmetu
Programiranje I

Pred vami je komplet nalog za utrjevanje snovi pri predmetu Programiranje I. Dodatnih nalog — za razliko od domačih — 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 so urejene po sklopih, znotraj posameznih sklopov pa (v grobem) po težavnosti. 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 I.

Poleg besedila nalog boste na spletni učilnici našli tudi paket, ki vsebuje testne primere in morebitne izhodiščne datoteke za posamezne naloge. Vsaka dodatna naloga (razen tistih v poglavju 12) je opremljena z desetimi testnimi primeri, s katerimi lahko vaše rešitve preizkusite. Če smo natančnejši:

- Pri nalogah tipa vhod-izhod imate na voljo 10 parov vhodnih datotek `vhod*.txt` in pripadajočih izhodnih datotek `izhod*.txt`. Testni primer *i* se šteje kot pravilno obravnavan natanko tedaj, ko vaš program pri branju vhoda iz datoteke `vhodi.txt` proizvede izpis, ki je enak vsebini datoteke `izhodi.txt`.
- Pri nalogah, ki se preverjajo s testnimi razredi, smo pripravili 10 parov testnih razredov (datotek `Test*.java`) in pripadajočih izhodnih datotek `izhod*.txt`. Vsak testni razred praviloma ustvari vsaj en objekt razreda, ki ga želite preizkusiti, 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 `Testi.java` dobimo izpis, ki je enak vsebini datoteke `izhodi.txt`.
- Pri nalogah na temo risanja v grafičnem oknu je na voljo 10 parov testnih razredov (datotek `Test*.java`) in pripadajočih izhodnih besedilnih datotek (`izhod*.txt`) oziroma slikovnih datotek (`izhod*.png`). Testni primeri s slikami se preverijo tako, da testni razred na podlagi vašega razreda ustvari slikovno datoteko, ta pa se nato po pikah primerja z referenčno. Medtem ko je posamezen besedilni testni primer lahko samo pravilen (1 točka) ali nepravilen (0 točk), je ocena slikovnega primera lahko poljubno realno število z intervala $[0, 1]$.

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

Veliko užitkov pri reševanju nalog!

1	Osnovni konstrukti I	1
1.1	Absolutna vrednost	1
1.2	Trihotomija	2
1.3	Časovna razlika I	2
1.4	Časovna razlika II	3
1.5	Najbližji večkratnik	3
1.6	Mediana trojice I	4
1.7	Urejanje trojice	5
2	Osnovni konstrukti II	7
2.1	Zaporedje zvezdic	7
2.2	Poštevanka I	8
2.3	Poštevanka II	8
2.4	Poštevanka III	9
2.5	Številska zaporedja	10
2.6	Potenca	12
2.7	Delitelji	13
2.8	EvroŠop [®]	13
2.9	Smučanje	14
2.10	Vozni red	15
3	Osnovni konstrukti III	17
3.1	Pari števil	17
3.2	Piramida števil	18
3.3	Igorjevi bloki	19
3.4	Šahovnica	20
3.5	Anžetove ledene sveče	22
3.6	Metaprogram	23
3.7	Razbijanje števil	24
4	Metode	27
4.1	Predvolilni golaž	27
4.2	Množenje z zaporednim seštevanjem	29
4.3	Mediana trojice II	29
4.4	Štetje klicev I (★)	30

4.5	Potenca po modulu (\star)	31
5	Naključja	33
5.1	Najboljše seme	33
5.2	Vraževerni Boris	34
5.3	Zdolgočasena Mojca	36
5.4	Šahovski popoldnevi	37
6	Razredi in objekti	39
6.1	Razreda Posta in Pismo	39
6.2	Razred Ulomek	42
6.3	Razred Datum	44
6.4	Dopolnitve razreda Oseba (\star)	46
7	Tabele I	49
7.1	Najbližji element	49
7.2	Digitalne črtice	50
7.3	Pascalov trikotnik	51
7.4	Telefonski imenik	52
7.5	Zlata sredina	53
7.6	Vsi različni I	54
7.7	Vsi različni II	54
7.8	Izstopajoči element	55
7.9	Kombinacije (\star)	57
7.10	Politična nasprotja I (\star)	58
7.11	Štetje klicev II (\star)	59
8	Tabele II	61
8.1	Maksimumi po stolpcih I	61
8.2	Maksimumi po stolpcih II	62
8.3	Pravilni trikotniki	62
8.4	Leksikografsko urejanje	64
8.5	Šahovski turnir	65
8.6	Determinanta	66
8.7	Politična nasprotja II (\star)	68
8.8	Tabela s poljubnim številom dimenzij (\star)	69
9	Tabele in objekti	73
9.1	Štiri v vrsto	73
9.2	Obedujoči filozofi	75
9.3	Krožki	78
9.4	Praštevila	82
9.5	Dopolnitve naloge <i>Prijatelji</i>	83
9.6	Poti po grafu (\star)	84
10	Dedovanje	87
10.1	Poštna pošiljke	87
10.2	Geometrijska telesa	89
10.3	Seznam (\star)	93
10.4	Naravno število (\star)	96
11	Grafika	99
11.1	Naloženi stolpični diagram	99

11.2 Barvni krog	101
11.3 Kvadratna spirala	101
11.4 Radarski diagram	104
11.5 Drevo (★)	105
11.6 Kochova snežinka (★)	107
12 Grafični uporabniški vmesnik	109
12.1 Pretvornik med številskimi sistemi	109
12.2 Diagram temperatur	110
12.3 Minolovec (★)	112
12.4 Elektronska preglednica (★)	117

Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

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

¹Pravila, 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`.

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.

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 $m_1 \leq m_2$.

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 $m_1 \leq m_2$.

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

Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

2.1 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 `*`). Izhis seveda tudi tokrat zaključite s skokom v naslednjo vrstico.

Testni primer 1

Vhod:

5

Izhod:

2.2 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
```

2.3 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 poštevanka 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
```

2.4 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
```

2.5 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 $a \leq b$ 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 pogojev:

- 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
```

2.6 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 $a^b \leq 10^9$.

Izhod

Izpišite vrednost potence a^b .

Testni primer 1

Vhod:

```
3 4
```

Izhod:

```
81
```

2.7 Delitelji

Naloga

Napišite program, ki prebere število in izpiše vse njegove delitelje.

Vhod

Na vhodu je zapisano celo število z intervala $[1, 10^6]$.

Izhod

Izpišite delitelje prebranega števila v naraščajočem vrstnem redu. Vsak delitelj izpišite v svoji vrstici.

Testni primer 1

Vhod:

30

Izhod:

1
2
3
5
6
10
15
30

2.8 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.

2.9 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
```

2.10 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 $m_z \leq m_k$.

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

Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

3.1 Pari števil

Naloga

Napišite program, ki prebere število n in izpiše vse pare števil od 1 do n , pri katerih je drugo število v paru strogo večje od prvega.

Vhod

Na vhodu je podano zgolj celo število $n \in [2, 50]$.

Izhod

Vsak par izpišite v svoji vrstici, števili v paru pa ločite s presledkom. Pare izpišite v leksikografskem vrstnem redu — torej naraščajoče po prvem številu v paru, v primeru enakih prvih števil pa naraščajoče po drugem številu v paru.

Testni primer 1

Vhod:

```
5
```

Izhod:

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

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

3.2 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
123456789012345678901

```

3.3 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
_7777777
```

Testni primer 2

Vhod:

```
2 2 3
```

Izhod:

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

3.4 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:

```

+ _ _ _ _ _ _ _ _ _ _ +
| _ _ _ _ _ * _ _ _ _ _ * _ _ _ _ _ |
| _ _ _ _ _ * _ _ _ _ _ * _ _ _ _ _ |
| _ * _ _ _ _ _ * _ _ _ _ _ * _ _ |
| _ * _ _ _ _ _ * _ _ _ _ _ * _ _ |
| _ _ _ _ _ * _ _ _ _ _ * _ _ _ _ _ |
| _ _ _ _ _ * _ _ _ _ _ * _ _ _ _ _ |
| _ * _ _ _ _ _ * _ _ _ _ _ * _ _ |
| _ * _ _ _ _ _ * _ _ _ _ _ * _ _ |
+ - - - - - - - - - - +

```

```

|uuuuu*_*uuuuu*_*uuuuu|
|uuuuu*_*uuuuu*_*uuuuu|
|_*_*uuuuu*_*uuuuu*_*_*|
|_*_*uuuuu*_*uuuuu*_*_*|
+u-u-u-u-u-u-u-u-u-u+

```

3.5 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:

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

3.6 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 fiksno n pa lahko nalogo rešimo s pomočjo n vgnezdjenih zank. Ker program seveda ne more vsebovati spremenljivega števila zank, boste napisali *metaprogram* — program, ki izpiše program, ki opisani problem reši s pomočjo n vgnezdjenih 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".
```

3.7 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 števkou v številu m . Dolžina prvega dela je tako enaka prvi števkou števila m , dolžina drugega dela je enaka drugi števkou števila m itd.

Nalogo rešite zgolj z operacijami nad celimi števili. Uporaba realnoštevilske operacije 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
```


Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

4.1 Predvolilni golaž

Naloga

Politiku Gvidu¹ 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 $[1, 10^3]$);

¹Po istoimenski skladbi Iztoka Mlakarja

- 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).

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.

4.2 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.

Testni primer 1

Vhod:

6 7

Izhod:

42

4.3 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 vrmeta 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.

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

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

Testni primer 1

Vhod:

7 10 8

Izhod:

8

Testni primer 2

Vhod:

5 3 5

Izhod:

5

4.4 Štetje klicev I (★)**Naloga**

Podani sta celi števili $a \geq 2$ in $b \geq 2$. Funkcija $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ 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.

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

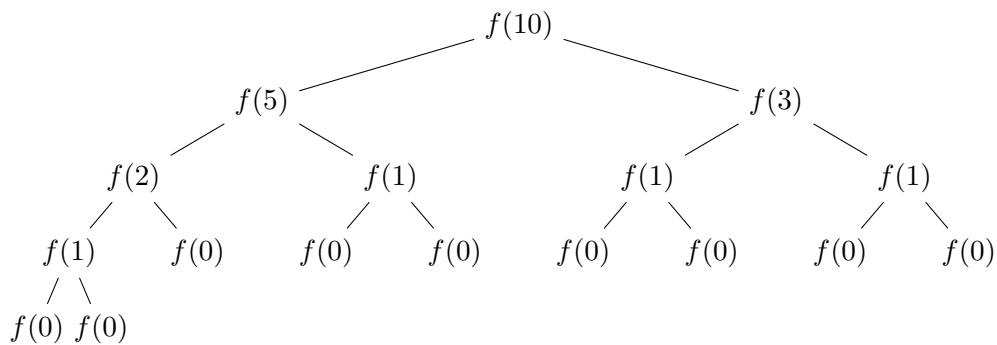
Vhod:

2 3 10

Izhod:

17

V tem primeru se funkcija f pokliče 17-krat:



4.5 Potenca po modulu (\star)

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 $a^{\lfloor b/2 \rfloor}$?

Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

5.1 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
```

5.2 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

$$D_{\sqcup}(T) : \sqcup M_{\sqcup}[S]$$

pri čemer je

- D zaporedna številka dneva, zapisana na 4 mesta (npr. $\sqcup\sqcup\sqcup 3$ ali $\sqcup 735$);
- 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.

5.3 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: k ocke | vsota = V

kjer je M zaporedna številka meta, k ocke 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.

5.4 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

$D.\text{dan}:\text{zaporedje}(N)$

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.

Splošna navodila

V tem sklopu nalog uporabljamo nekoliko drugačen pristop kot pri prejšnjih. Pri vseh nalogah zahtevamo realizacijo nekega razreda in njegovih javno dostopnih konstruktorjev in metod. Glede ostalih komponent (atributi, morebitne privatne metode oz. konstruktorji itd.) ste svobodni, seveda pa upoštevajte splošne smernice objektno usmerjenega programiranja. Svoje rešitve boste lahko preverili z množico testnih razredov in pripadajočih izhodnih datotek. Testni razredi so zasnovani tako, da učinkovitost vaše rešitve ni bistven dejavnik. Glavno merilo naj bo tokrat »eleganca« rešitve, ne njena optimalnost v smislu prostorsko-časovne učinkovitosti. Pri elegantni rešitvi se, denimo, metode »rade« kličejo med seboj.

6.1 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 pošto 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 pošto številko in nazivom.

- `public int vrniStevilko()`

Vrne pošto 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 izvirno 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 izvirno in ciljno pošto, »priporočenostjo« (true: priporočeno; false: navadno) ter razdaljo (v kilometrih) med izvirno 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` izvirna pošta za pismo `this`.

- `public boolean staIzvorInCiljIsta()`

Vrne `true` natanko v primeru, če ima pismo `this` isto izvirno 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

vklučno $(3r - 1)$ 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]
```

6.2 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!)
```

6.3 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.

6.4 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 očetovskaGeneracijskaRazlika(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:

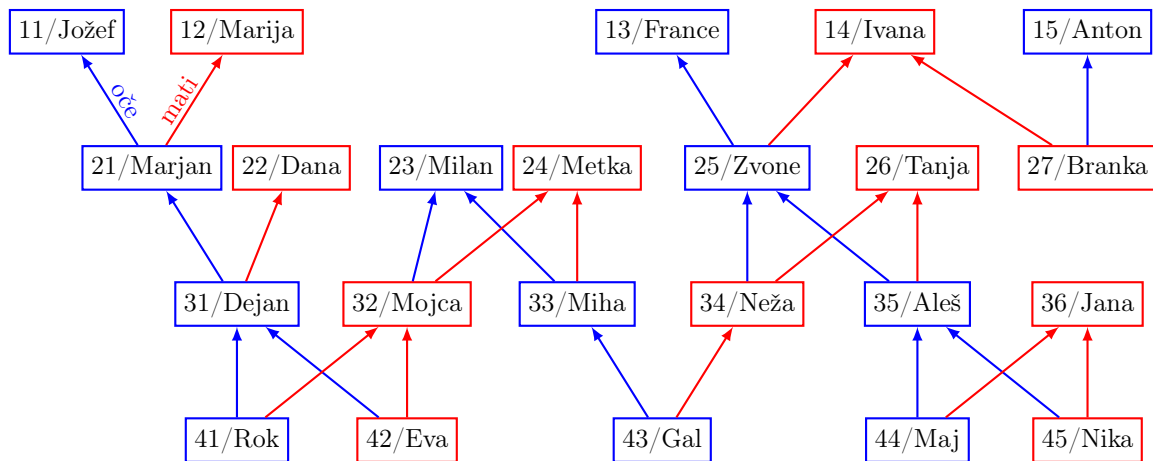
- $A = B$ (vsaka oseba je sam svoj prednik).
- A je prednik B -jevega očeta.
- A je prednik B -jeve matere.

- `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



Slika 6.1: Starševski odnosi med osebami v testnem razredu.

uporabi za posamezne prednike. Za primer s slike 6.1 bi klic `os43.nasteljPrednike()` 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 6.1) 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 {

    public static void main(String[] args) {
        Oseba os11 = new Oseba("Jožef", "Pogačnik", 'M', 1921);
        Oseba os12 = new Oseba("Marija", "Pogačnik", 'Z', 1928);
        Oseba os13 = new Oseba("France", "Kotnik", 'M', 1932);
        Oseba os14 = new Oseba("Ivana", "Kotnik", 'Z', 1931);
        Oseba os15 = new Oseba("Anton", "Zajc", 'M', 1922);
        Oseba os21 = new Oseba("Marjan", "Pogačnik", 'M', 1946, os11, os12);
        Oseba os22 = new Oseba("Dana", "Pogačnik", 'Z', 1950);
```

```

Oseba os23 = new Oseba("Milan", "Smole", 'M', 1953);
Oseba os24 = new Oseba("Metka", "Smole", 'Z', 1953);
Oseba os25 = new Oseba("Zvone", "Kotnik", 'M', 1956, os13, os14);
Oseba os26 = new Oseba("Tanja", "Kotnik", 'Z', 1954);
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);
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.nasteljPrednike();
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

```


Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek.

7.1 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.

7.2 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 številke:

Številka	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.

7.3 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 $i \in \{1, \dots, n + 1\}$) 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

```

7.4 Telefonski imenik

Naloga

Napišite program, ki najprej prebere n osebnih imen in pripadajočih telefonskih števil, 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 števil. 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.

7.5 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
```

7.6 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
```

7.7 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
```

7.8 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.

7.9 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 $i \geq 2$) 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.

7.10 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
LCDDDCCL
DDDCCLL
DDDCCLL
DDDCCLL
DDCCLL
DDCCLL
DDCCLL
DCLLCDD
DCDDCCL
CLLCDDD
CDDDCCL

Testni primer 2

Vhod:

2 3 1

Izhod:

2

V tem primeru sta možni le razporeditvi LLCDD in DDDCLL.

7.11 Štetje klicev II (★)

Naloga

Podani sta celi števili $a \geq 2$ in $b \geq 2$. Funkcija $f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ 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

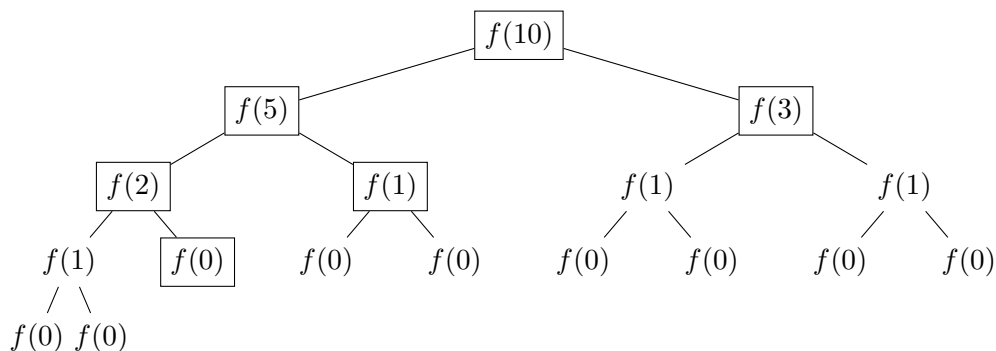
Vhod:

2 3 10

Izhod:

6

V tem primeru imamo 6 različnih klicev funkcije f :



Splošna navodila

Pri nalogah 8.1–8.7 lahko svoje rešitve preverite z množico vhodnih datotek in pripadajočih izhodnih datotek, pri nalogi 8.8 pa z množico testnih razredov in pripadajočih izhodnih datotek.

8.1 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]
```

8.2 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.

8.3 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 pravičnega, 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 pravičnih 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 pravične trikotnike. Vsaka trojica naj se izpiše samo enkrat, in sicer v obliki

$i \sqcup j \sqcup k$

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 pravičnega 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
```

8.4 Leksikografsko urejanje

Naloga

Vektor $(a_0, a_1, \dots, a_{n-1})$ je *leksikografsko manjši* od vektorja $(b_0, b_1, \dots, b_{n-1})$, če obstaja indeks $i \geq 0$, 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]
```

8.5 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 \square *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
```

8.6 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
```

8.7 Politična nasprotja II (★)

Naloga

Opomba: Ta naloga je enaka nalogi *Politična nasprotja* iz poglavja 7, 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
LCDDDC
```

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.

8.8 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 $[a_0, a_1, \dots, a_{d_1-1}]$, kjer so $a_0, a_1, \dots, a_{d_1-1}$ elementi p-tabele. P-tabela velikosti $d_1 \times d_2 \times d_3 \times \dots \times d_n$ naj bo predstavljena z nizom $[S_0, S_1, \dots, S_{d_1-1}]$, kjer so $S_0, S_1, \dots, S_{d_1-1}$ nizi, ki predstavljajo posamezne pod-p-tabele velikosti $d_2 \times d_3 \times \dots \times d_n$.

Testni primer 8

Testni razred:

```
import java.util.Random;

public class Test08 {

    public static void main(String[] args) {
        // delali bomo s tabelo 3 x 4 x 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: Zavaljo 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?

Splošna navodila

Pri nalogi 9.6 lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek, pri ostalih nalogah pa z množico testnih razredov in pripadajočih izhodnih datotek. Pri nalogi 9.5 izhajajte iz razredov, ki jih boste našli v mapi s testnimi datotekami.

Pri nalogah, ki zahtevajo izdelavo razreda, so navedeni zgolj javno dostopni elementi, ki jih mora razred vsebovati. Seveda pa lahko po potrebi dodajate tudi svoje (privatne in javne) elemente.

9.1 Štiri v vrsto

Naloga

Pri igri Štiri v vrsto igralca izmenično spuščata žetone (na primer, 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 igravec)`

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

- `public int izid()`

Če je prvi igravec postavil najmanj štiri zaporedne žetone v neki vrstici, stolpcu ali diagonalni, drugemu pa to ni uspelo, naj metoda vrne vrednost 0. Če se s tem dosežkom lahko ponaša drugi igravec, 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
    }
}
```

```

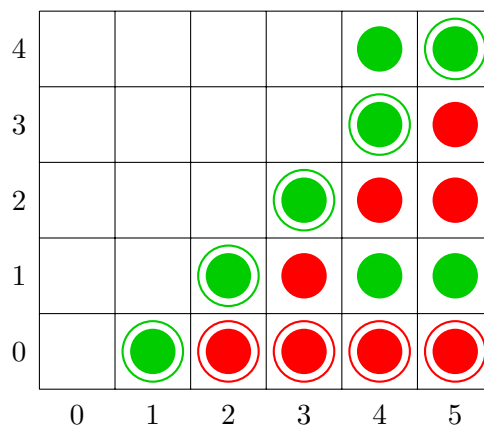
System.out.println(igra.vrzi(4));      // false
System.out.println(igra.vrzi(5));      // true
System.out.println(igra.naPotezi());   // 1
System.out.println(igra.vrzi(5));      // true
System.out.println(igra.naPotezi());   // 0

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 9.1 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`.



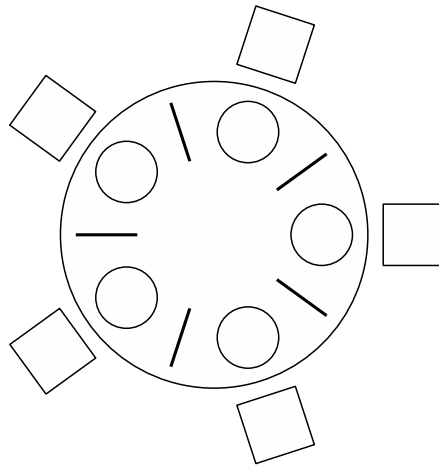
Slika 9.1: Vsebina igralnega okvirja po koncu izvajanja kode v testnem razredu 9.

9.2 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 9.2). 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 sosеду, da bi pričela s prehranjevanjem. Ko filozof preneha jesti, odloži (in s tem sprostí) obe paličici.¹

¹Gre 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 obedujočih filozofov tako ilustrira borbo procesov za omejene vire.



Slika 9.2: 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():**

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
    }
}
```

```

    izpisiSosedo(mladen);          // Renata/Tine
    izpisiSosedo(tine);            // Mladen/Spomenka
    izpisiSosedo(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 izpisiSosedo(Filozof filozof) {
    Filozof[] sosedo = filozof.vrniSosedo();
    System.out.printf("%s/%s%n", sosedo[0].vrniIme(), sosedo[1].vrniIme());
}
}

```

9.3 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 **Učenec** in **Krozek** tako, da bodo njuni objekti predstavljali posamezne učence oziroma krožke. Razred **Učenec** 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 Učenec(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());
    }
}
}

```

9.4 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
    }
}

```

9.5 Dopolnitve naloge *Prijatelji*

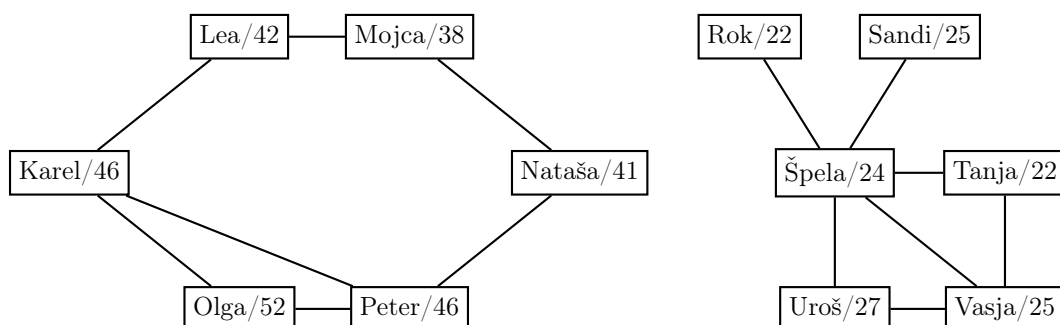
Naloga

Razred `Oseba`, ki smo ga napisali na vajah in ki ga najdete tudi v mapi s testnimi datotekami za to nalogo, dopolnite s sledečimi metodami:

- `public int steviloPrijateljevPrijateljev():`

Vrne število prijateljev prijateljev osebe `this`, ki niso obenem neposredni prijatelji osebe `this`. Tudi osebe `this` ne štejemo med prijatelje prijateljev osebe `this`.

Slika 9.3 prikazuje primer množice oseb (za vsako osebo je podano njeno ime in starost) in prijateljstev med njimi. Osebi sta na sliki povezani natanko tedaj, ko sta prijatelja. V tem primeru so vsa prijateljstva vzajemna. Karel ima dva prijatelja prijateljev: Mojco (preko Lee) in Natašo (preko Petra). Peter je Karlov neposredni prijatelj, zato ga ne štejemo poleg. Uroš ima tri prijatelje prijateljev: Roka, Sandija in Tanjo.



Slika 9.3: Primer vzajemnih prijateljstev.

- `public Oseba[] prijateljiPoStarosti():`

Vrne novo tabelo, v kateri so prijatelji osebe `this` padajoče urejeni po starosti. Če je več prijateljev enako starih, naj bodo ti v tabeli urejeni po naraščajočem abecednem vrstnem redu glede na »IP-je« (imena in priimke). Za primerjanje nizov po abecedi uporabite metodo `compareTo` iz razreda `String`.

V primeru na sliki 9.3 bi metoda za osebo Špela vrnila tabelo, v kateri bi bili v tem vrstnem nanizane sledeče osebe: Uroš, Sandi, Vasja, Rok in Tanja.

- `(★) public static boolean[][] povezanost(Oseba[] osebe):`

Definirajmo relacijo *povezanosti* takole: osebi A in B sta povezani, če velja $A = B$ ali pa če obstaja oseba C , tako da je A prijatelj osebe C ali C prijatelj osebe A , osebi C in B pa sta povezani. (Učeno pravimo, da je relacija povezanosti *refleksivno-simetrično-transitivna ovojnica* relacije prijateljstva.)

Metoda `povezanost` naj vrne matriko velikosti $n \times n$ (kjer je n dolžina tabele `osebe`), v kateri ima element (i, j) vrednost `true` natanko v primeru, če sta osebi i in j povezani.

V primeru na sliki 9.3 bi metoda `povezanost` za tabelo {Karel, Lea, Mojca, Nataša, Olga, Peter, Rok, Sandi, Špela, Tanja, Uroš, Vasja} vrnila sledečo matriko ($1 \rightarrow \text{true}$, $0 \rightarrow \text{false}$):

```

1 1 1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1

```

9.6 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 $v \in [0, n-1] \setminus \{u\}$, 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 $v_k \in [0, n-1] \setminus \{v_z\}$, 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 $u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_k$ naj se torej izpiše pred potjo $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_l$ natanko tedaj, ko je izpolnjen eden od sledečih pogojev:

- $k < l$ in $u_i = v_i$ za vse $i \in \{1, \dots, k\}$;
- obstaja $i \in \{1, \dots, k\}$, 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

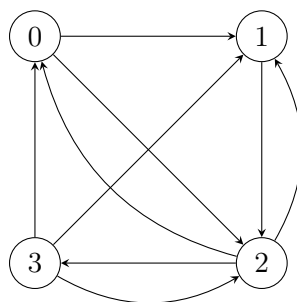
V tem testnem primeru iščemo poti od vozlišča 3 do vozlišča 1 v grafu na sliki 9.4.

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 9.4: Graf v testnem primeru 7.

Splošna navodila

Pri nalogah 10.1 in 10.2 lahko svoje rešitve preverite z množico vhodnih in pripadajočih izhodnih datotek, pri nalogah 10.3 in 10.4 pa z množico testnih razredov in pripadajočih izhodnih datotek.

10.1 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 poštних 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 pošilk na izhodu mora biti enak vrstnemu redu pošilk 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 pošilkah. V vsaki vrstici je najprej podana vrsta pošilke (**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)`.

10.2 Geometrijska telesa

Naloga

Na vnhodu so zapisani osnovni podatki o različnih geometrijskih telesih: kvadrilih, kockah in kroglih. 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 kvadrilih 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)

```

UUUUUUUU*U*U*
UUUUUUUU*U*U*
UUUUUUUU*U*U*
UUUUUUUU*U*U*
+U+U+U+UOUOUOU+U+U+U+
+U+U+U+UOUOUOU+U+U+U+
UUUUUUUU*U*U*
UUUUUUUU*U*U*
UUUUUUUU*U*U*
UUUUUUUU*U*U*
UUUUUUUUOUOUO
UUUUUUUUOUOUO

```

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
      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
=====

```

10.3 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 izdelava objekt, ki predstavlja prazen seznam. Razred **Neprazen** pa mora vsebovati konstruktor

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

ki izdelava 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

`[a1, □ a2, □ ... □ an]`

kjer so a_1, a_2, \dots, 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]
    }
}

```

10.4 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 `NoSuchElementException`.

- `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
    }
}
```

Splošna navodila

Pri vseh nalogah v tem sklopu lahko svoje rešitve preverite z množico testnih razredov in pripadajočih izhodnih besedilnih in slikovnih datotek. Izhajajte iz razredov, ki jih boste našli v mapah s testnimi datotekami. Razreda `Platno` ne spreminjajte.

11.1 Naloženi stolpični diagram

Naloga

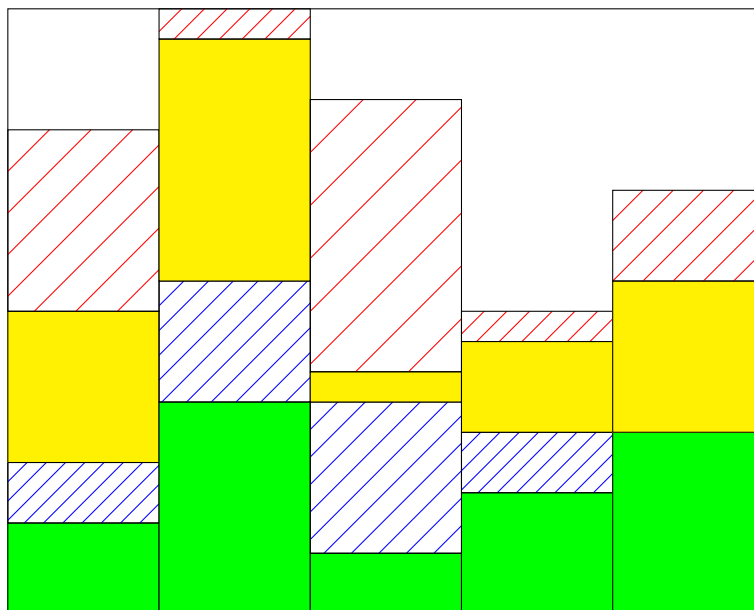
Metodo `narisi` v razredu `NalozeniStolpicniDiagram` dopolnite tako, da bo narisala naloženi stolpični diagram na podlagi tabel `this.podatki` (tipa `double[][]`) in `this.slogi` (tipa `Slog[]`). Na primer, za tabeli

```
podatki:
{
    { 3.0, 2.0, 5.0, 6.0 },
    { 7.0, 4.0, 8.0, 1.0 },
    { 2.0, 5.0, 1.0, 9.0 },
    { 4.0, 2.0, 3.0, 1.0 },
    { 6.0, 0.0, 5.0, 3.0 }
}

slogi:
{
    new Pobarvan(Color.GREEN),    // barva
    new Srafiran(10, Color.BLUE), // stCrt (≥ 1), barvaCrt
    new Pobarvan(Color.YELLOW),
    new Srafiran(7, Color.RED),
}
```

naj se nariše diagram, kot ga prikazuje slika 11.1.

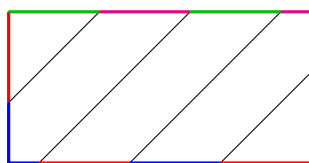
Podatek v i -ti vrstici in j -tem stolpcu tabele `podatki` naj se prikaže kot j -ti pravokotnik (gledano od spodaj navzgor) v i -tem stolpcu diagrama (gledano od leve proti desni). Di-



Slika 11.1: Naloženi stolpični diagram (testni primer 4).

agram naj se po širini razteza čez celotno platno. Višina najvišjega stolpca naj bo enaka višini platna. Vsi pravokotniki naj bodo enako široki, višina posameznih pravokotnikov pa naj bo premosorazmerna pripadajočim vrednostim v tabeli **podatki**.

Tabela **slogi** podaja sloge posameznih pravokotnikov znotraj istega stolpca. Prvi element tabele podaja slog prvega (najbolj spodnjega) pravokotnika v vsakem stolpcu, drugi element podaja slog drugega pravokotnika itd. Slog je bodisi *pobarvan* (objekt tipa **Pobarvan**) bodisi *šrafiran* (objekt tipa **Srafiran**). Pravokotnik s slogom *pobarvan* je treba pobarvati z barvo, ki jo določa atribut **barva**. V notranjosti pravokotnika s slogom *šrafiran* pa je treba narisati **stCrt** enakomerno razmaknjenih črt barve **barvaCrt** pod kotom 45° . Za lažje razumevanje šrafure si oglejte sliko 11.2. Pravokotniki obeh slogov naj imajo obrobo barve **Color.BLACK**.



Slika 11.2: Pravokotnik, šrafiran s štirimi črtami. Vsi odseki na obodu pravokotnika so enako dolgi. Odseki so namenjeni le lažjemu razumevanju šrafure; obroba pravokotnika naj bo v celoti črna.

Poleg metode **narisi** dopolnite še sledeči metodi:

- **public double sirinaStolpca(double wp, double hp)**

Vrne širino stolpca pri podani širini (**wp**) in višini (**hp**) platna.

- **public double visinaEnote(double wp, double hp)**

Vrne višino pravokotnika, ki ustreza podatku z vrednostjo 1.0.

Tabela **podatki** je pravokotna tabela velikosti $n \times m$ ($n \in [1, 100]$, $m \in [1, 100]$), tabela **slogi** pa vsebuje m elementov. Vsi elementi tabele **podatki** so ne-negativni.

11.2 Barvni krog

Naloga

Metodo `narisi` v razredu `BarvniKrog` dopolnite tako, da bo narisala krog, sestavljen iz n ($n \geq 2$) enako velikih zaporednih krožnih izsekov. Krožni izseki naj bodo napolnjeni z barvami, ki so v modelu HSB (Hue, Saturation, Brightness) določene takole:

- komponenta S naj bo pri vseh krožnih odsekih enaka 1 (maksimalna možna vrednost);
- komponenta B naj bo pri vseh krožnih odsekih enaka 1 (maksimalna možna vrednost);
- komponenta H naj bo pri prvem krožnem izseku (izseku, ki se prične pri kotu 0°) enaka 0, pri vsakem naslednjem pa naj bo za $1/n$ večja.

Krog naj zavzema celotno krajšo stranico platna, po daljši stranici pa naj bo prikazan na sredini.

Število n se ob klicu konstruktorja razreda `BarvniKrog` prenese v atribut `stIzsekov`.

Primeri

Slika 11.3 prikazuje vsebino platna za $n = 2$, $n = 3$, $n = 4$, $n = 5$, $n = 10$ in $n = 1000$.

Napotki

- Pomagajte si z metodo `getHSBColor` iz razreda `java.awt.Color`.
- Metoda `g.fillArc`, kjer je `g` objekt tipa `Graphics2D`, sprejme samo celoštevilске kote, kar je predvsem pri večjem številu izsekov premalo za natančen izris. Zato izseke raje rišite takole:

```
g.fill(new Arc2D.Double(..., ..., ..., ..., ..., ..., Arc2D.PIE));
```

Prvih šest parametrov konstruktorja razreda `Arc2D.Double` je enakih kot pri metodi `g.fillArc`, le da so lahko tipa `double`.

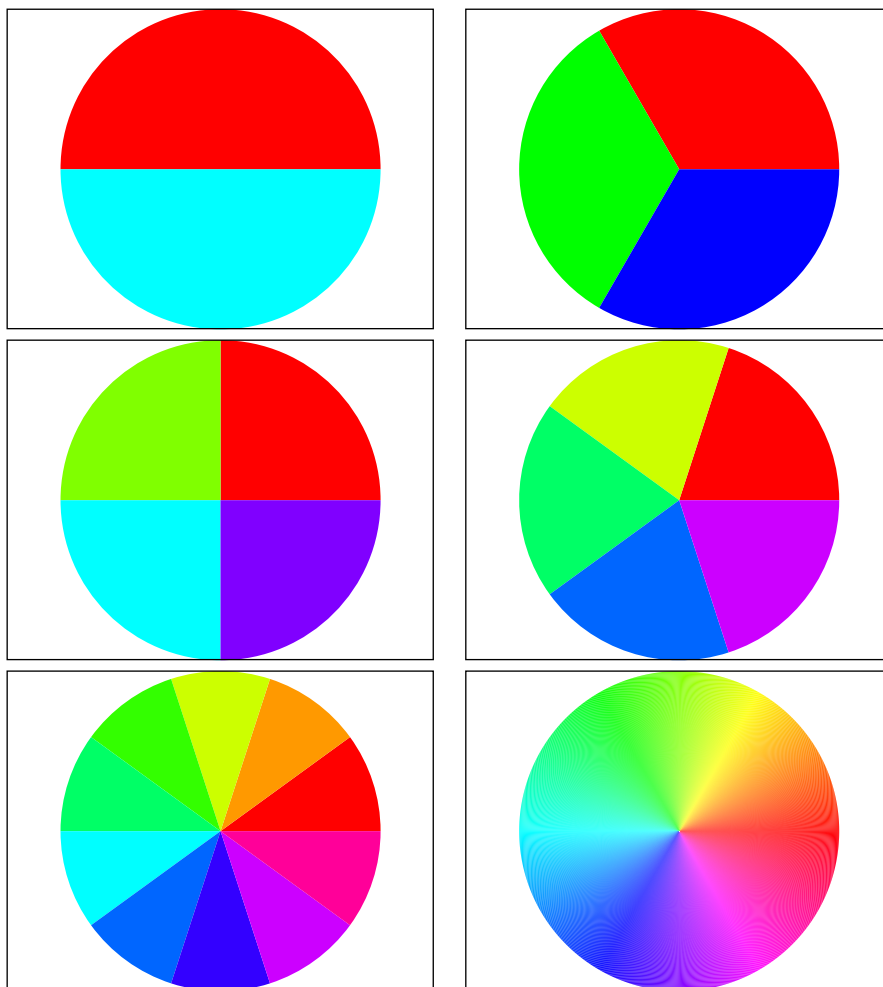
11.3 Kvadratna spirala

Naloga

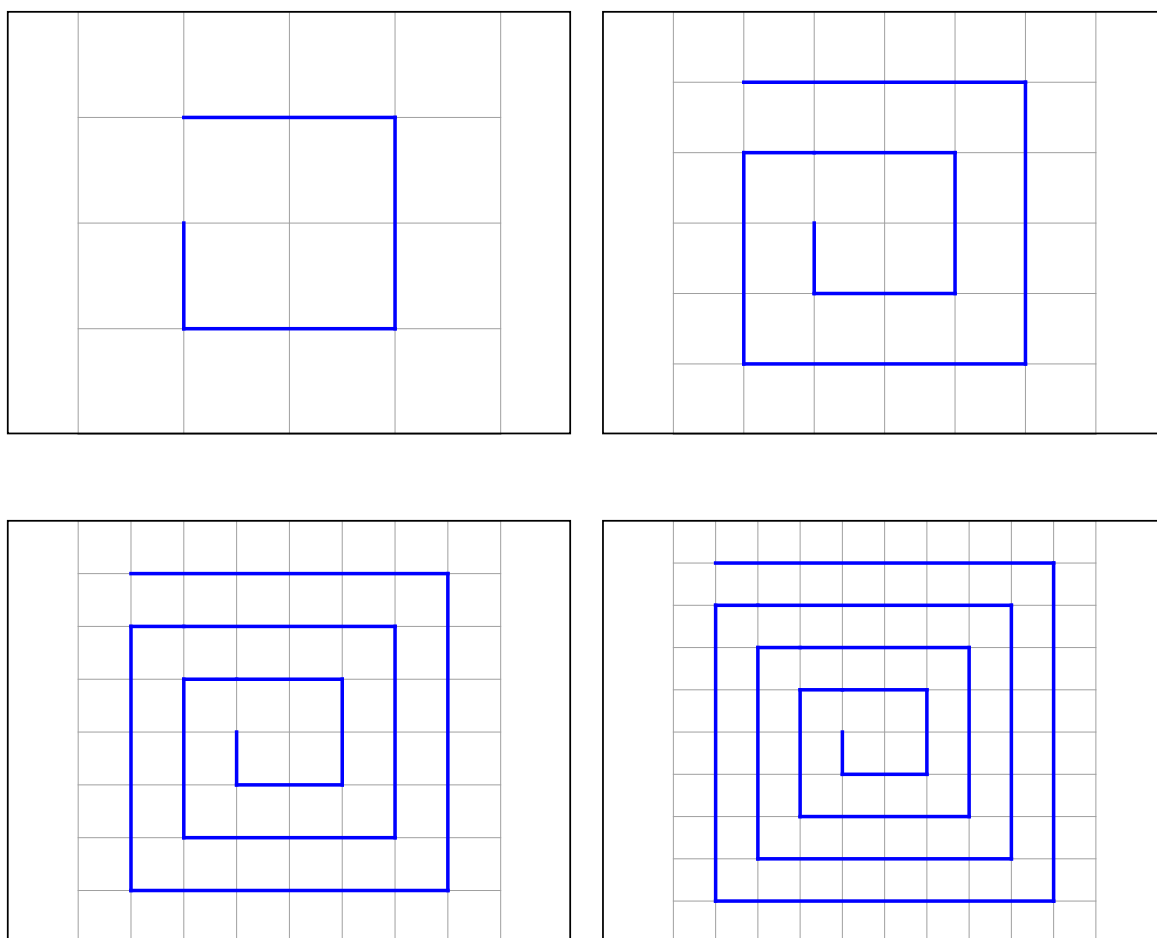
Metodo `narisi` v razredu `Spirala` dopolnite tako, da bo narisala kvadratno spiralo, sestavljeno iz n polnih zavojev. Slika 11.4 prikazuje vsebino platna za $n = 1$, $n = 2$, $n = 3$ in $n = 4$. Kvadratne mreže ne rišite, saj je prikazana zgolj zato, da vam olajša določitev ključnih mer. Mreža zavzema celotno krajšo stranico platna, po daljši stranici pa je postavljena na sredino.

Spiralo narišite z barvo `Color.BLUE`. Uporabite privzeto debelino črt; črte na spodnji sliki so odebeljene zgolj zaradi boljše vidnosti.

Število n se ob klicu konstruktorja razreda `Spirala` prenese v atribut `stZavojev`.



Slika 11.3: Barvni krogi za različne vrednosti n .



Slika 11.4: Kvadratna spirala za različne vrednosti n .

11.4 Radarski diagram

Naloga

Metodo `narisi` v razredu `RadarskiDiagram` dopolnite tako, da bo narisala *radarski diagram* na podlagi tabele `this.delezi` tipa `double[]`. Na primer, za tabelo

`delezi: {0.6, 0.3, 0.8, 0.5, 0.9}`

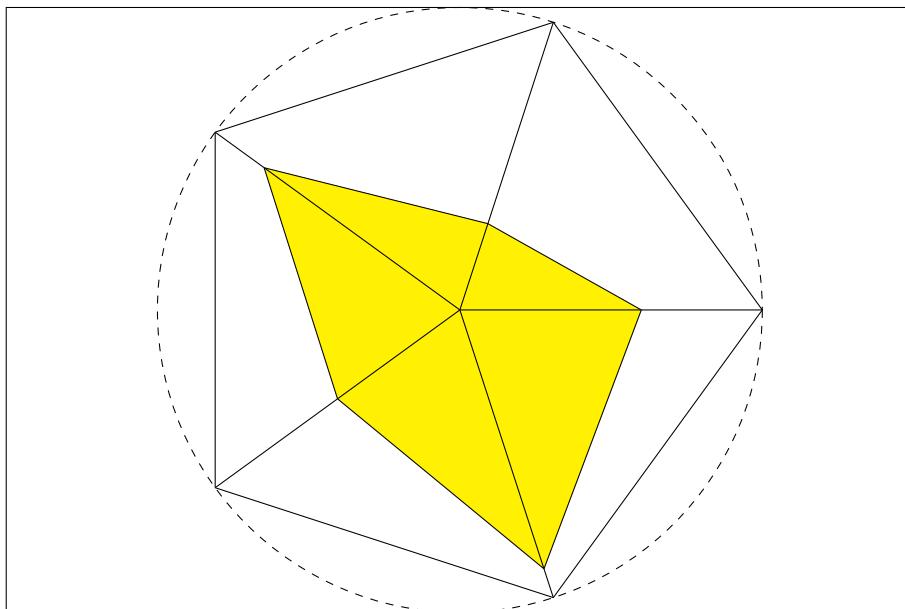
naj se nariše diagram, kot ga prikazuje slika 11.5.

Radarski diagram je sestavljen iz dveh enako orientiranih n -kotnikov (n je dolžina tabele `delezi`) in daljic, ki povezujejo središče platna s posameznimi oglišči zunanjega n -kotnika. Zunanji n -kotnik je včrtan navideznemu krogu, ki leži na sredini platna in zavzema celotno krajšo stranico platna. Zunanji n -kotnik je pravilen, eno od njegovih oglišč pa se nahaja pri kotu 0, torej na skrajni desni točki navideznega kroga.

Oglišča notranjega n -kotnika so določena z elementi tabele `delezi`, in sicer tako, da oglišče, ki pripada elementu z vrednostjo r , razdeli daljico med središčem platna in ogliščem zunanjega n -kotnika v razmerju $r : (1 - r)$, gledano od središča proti oglišču. Prvemu elementu tabele `delezi` pripada oglišče pri kotu 0, nato pa si oglišča sledijo v nasprotni smeri urinega kazalca.

Notranji n -kotnik pobarvajte z barvo `Color.YELLOW`. Oba n -kotnika obrobite z barvo `Color.BLACK`. S to barvo narišite tudi daljice od središča platna do oglišč zunanjega n -kotnika.

Tabela `delezi` vsebuje vsaj 3 in največ 100 elementov, posamezni elementi pa pripadajo intervalu $(0.0, 1.0]$.



Slika 11.5: Radarski diagram za podatke $\{0.6, 0.3, 0.8, 0.5, 0.9\}$ (testni primer 1).

Namiga

- Če ima krog polmer r in središče (x_c, y_c) , kako izračunamo koordinati točke na obodu kroga pri kotu φ ?

- Pomagajte si z metodama `drawPolygon` in `fillPolygon` iz razreda `Graphics`.

11.5 Drevo (★)

Naloga

Metodo `narisi` v razredu `Drevo` dopolnite tako, da bo narisala drevo, predstavljeno s podanim opisom. Opis drevesa je neprazen niz, sestavljen iz ne-negativnih celih števil, ki so med seboj ločena s po enim presledkom. Na začetku opisa je podano število otrok korena drevesa, nato pa so na enak način (rekurzivno) po vrsti opisani posamezni otroci. Na primer, opis

4 2 0 0 1 2 0 0 0 3 0 3 0 0 0 0

predstavlja drevo, v katerem ima koren 4 otroke. Prvi otrok korena ima dva otroka; oba od njiju sta brez otrok. Drugi otrok korena ima enega otroka, ta pa ima dva otroka; oba od njiju sta brez otrok. Tretji otrok korena nima nobenega otroka. Četrty otrok korena ima tri otroke: prvi je brez otrok, drugi ima tri otroke (vsi so brez otrok), tretji pa je prav tako brez otrok. Za lažje razumevanje bomo dele opisa, ki pripadajo posameznim poddrevesom korena, označili z različnimi barvami:

4 2 0 0 1 2 0 0 0 3 0 3 0 0 0 0

Drevo naj po višini in širini zavzema celotno površino platna. Vsakemu vozlišču naj pripada (navidezna) pravokotna celica. Vse celice naj bodo enako visoke, njihove širine pa so določene s sledečima praviloma:

- Celica s korenem drevesa zavzema celotno širino platna.
- Če ima neko vozlišče drevesa n otrok, potem je širina celice vsakega njegovega otroka enaka $1/n$ širine celice starševskega vozlišča.

Vozlišča naj bodo prikazana kot enako veliki polni krogi v središčih posameznih celic. Premer vozlišča naj bo enak polovici širine oz. višine (vzame naj se manjša od obeh dimenzij) najožje celice drevesa. Vozlišča naj bodo brez obrobe, zapolnjena pa naj bodo z barvo `Color.BLUE`. Isto barvo uporabite za povezave.

Lahko predpostavite, da je opis drevesa pravilen. Obrobe celic, ki so prikazane na sledečih vzorčnih slikah, so namenjene zgolj lažji predstavi, zato jih ne rišite.

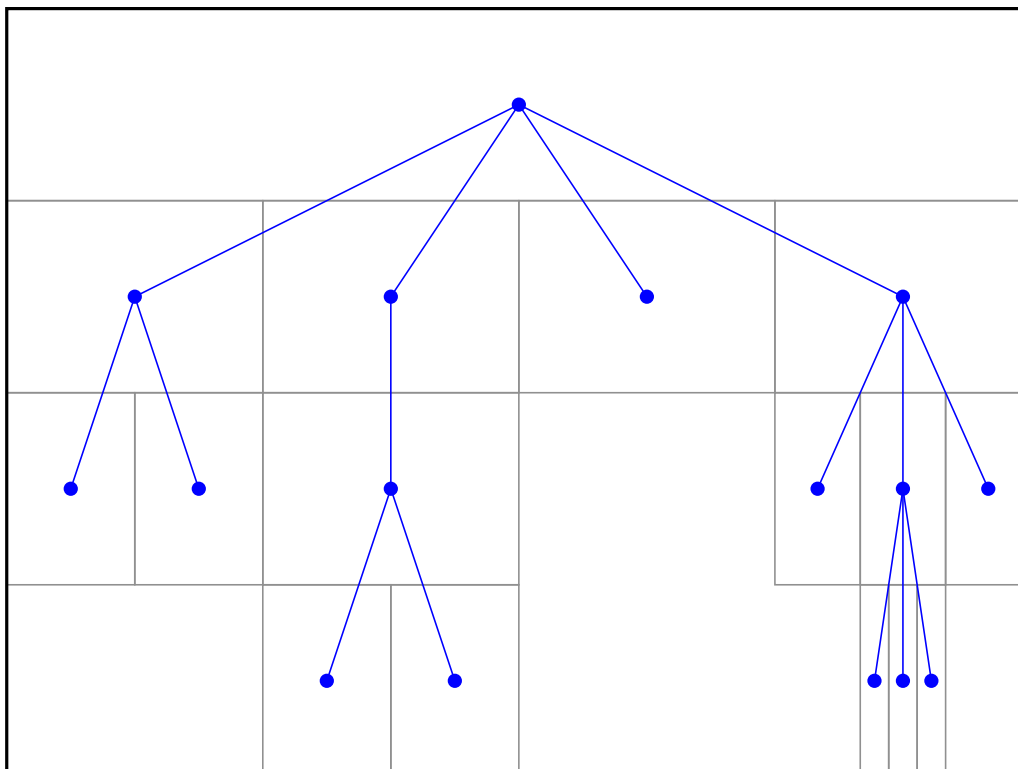
Opis drevesa se ob klicu konstruktorja razreda `Drevo` prenese v atribut `opis`.

Primeri

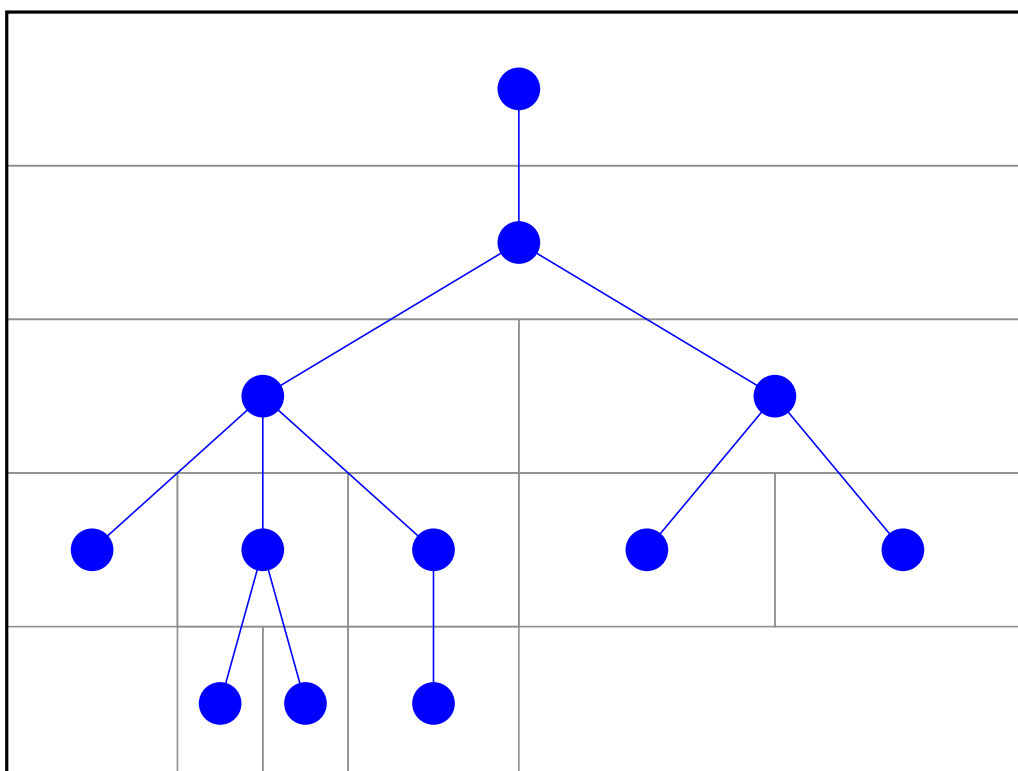
Sliki 11.6 in 11.7 prikazujeta drevesi iz testnih primerov 1–3 in 4.

Napotek

Opis drevesa lahko na posamezne komponente razbijete s pomočjo metode `split` iz razreda `String`. Druga možnost pa je, da opis drevesa podate konstruktorju razreda `Scanner` in se potem z objektom tega tipa »sprehodite« po nizu na enak način, kot bi se po standardnem vhodu.



Slika 11.6: Drevo z opisom 4 2 0 0 1 2 0 0 0 3 0 3 0 0 0 0.

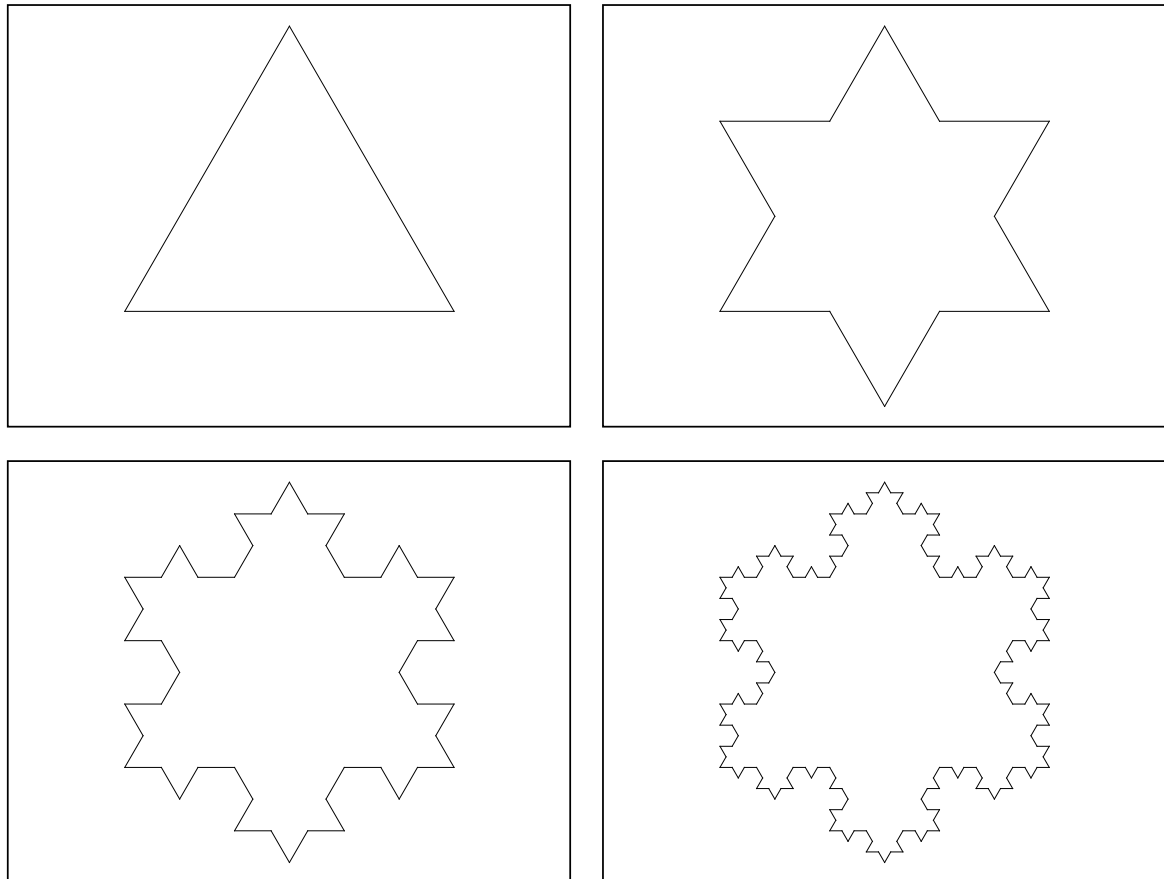


Slika 11.7: Drevo z opisom 1 2 3 0 2 0 0 1 0 2 0 0.

11.6 Kochova snežinka (★)

Naloga

Metodo `narisi` v razredu `KochovaSnezinka` dopolnite tako, da bo narisala *Kochovo snežinko* po n iteracijah ($n \geq 1$). Kochova snežinka po prvi iteraciji je enakostranični trikotnik, vsako naslednjo iteracijo pa dobimo tako, da vsako daljico iz trenutne iteracije razdelimo na tri enake dele, nad srednjim delom narišemo enakostranični trikotnik, nato pa srednji del daljice pobrišemo. Slika 11.8 prikazuje Kochove snežinke po prvih štirih iteracijah.



Slika 11.8: Kochove snežinke po prvih štirih iteracijah.

V vašem programu Kochove snežinke ne boste risali po opisani iterativni shemi, pač pa boste kar neposredno (brez vmesnih iteracij) narisali snežinko po podanem številu iteracij. Še več: **snežinko narišite v eni potezi**, torej kot sklenjeno lomljenko. Pričnite v nekem oglišču snežinke, povlecite daljico do naslednjega oglišča, od tam povlecite daljico do naslednjega oglišča itd. Vse daljice narišite z barvo `Color.BLACK`.

Stranico enakostraničnega trikotnika, ki predstavlja Kochovo snežinko v iteraciji 1, določite s formulo

$$d = \frac{9}{10} \min(w_p, \frac{\sqrt{3}}{2} h_p),$$

kjer w_p in h_p označujeta širino in višino platna. Trikotnik naj bo po širini prikazan na sredini platna, vse nadaljnje Kochove snežinke pa naj ležijo na sredini platna tudi po višini.

Namig

Kochovo snežinko je možno razdeliti na tri enake dele, ki izvirajo iz posameznih stranic izhodiščnega enakostraničnega trikotnika. Ali lahko posamezni del iteracije i sestavite s pomočjo posameznih delov iteracije $i - 1$?

Splošna navodila

Pri nalogah v tem sklopu boste morali shajati brez testnih primerov in vnaprej pripravljenih ogrodi razredov.

12.1 Pretvornik med številskimi sistemi

Naloga

Napišite program za pretvarjanje med številskimi sistemi v grafičnem oknu. Program naj uporabniku omogoča vnos izvirnega in ciljnega številskega sistema ter pozitivnega celega števila, zapisanega v izvirnem številskem sistemu. Po uporabnikovem vnosu podatkov in pritisku na gumb naj program vnešeno število pretvori v podani ciljni številski sistem in prikaže rezultat v obliki oznake (`JLabel`). Na primer, če dvojiško število 11010011 pretvorimo v šestnajstiški sistem, dobimo rezultat D3. (Šestnajstiška števila so sestavljena iz števk 0, 1, ..., 9, A, B, C, D, E in F.)

Program naj deluje za številske sisteme od dvojiškega do vključno šestnajstiškega. Lahko predpostavite, da bo desetiška vrednost podanega števila vedno enaka kvečjemu `Integer.MAX_VALUE`. (Z drugimi besedami: vaš program ima »pravico«, da pri prevelikih vhodnih številih daje napačne rezultate.) Z izjemo te dopustne pomanjkljivosti pa naj program deluje karseda robustno: v primeru neveljavnega vnosa naj ne prikaže obvestila o neulovljeni izjemi, pač pa naj namesto rezultata izpiše niz **Napaka**.

Primeri

Na sliki 12.1 je prikazanih nekaj primerov izvajanja programa po vnosu podatkov in pritisku na gumb za pretvorbo. V četrtem primeru gre za napako v vhodnem številu (petiška števila lahko vsebujejo zgolj številke med 0 in 4), v petem pa za napako v ciljnem številskem sistemu (možne vrednosti so med 2 in 16).

The image shows six examples of a number conversion interface, arranged in two rows of three. Each example consists of a box with the following fields:

- Izvorni sistem:** Input field for the source base.
- Ciljni sistem:** Input field for the target base.
- Vhodno število:** Input field for the number to be converted.
- Pretvori:** A blue button to perform the conversion.
- Rezultat:** The output of the conversion.

Examples of conversions shown:

- Example 1: Izvorni sistem: 2, Ciljni sistem: 16, Vhodno število: 11010011, Rezultat: D3
- Example 2: Izvorni sistem: 7, Ciljni sistem: 5, Vhodno število: 63, Rezultat: 140
- Example 3: Izvorni sistem: 16, Ciljni sistem: 14, Vhodno število: C3B, Rezultat: 11D9
- Example 4: Izvorni sistem: 5, Ciljni sistem: 10, Vhodno število: 3504, Rezultat: Napaka
- Example 5: Izvorni sistem: 11, Ciljni sistem: 17, Vhodno število: 496, Rezultat: Napaka

Slika 12.1: Primeri delovanja pretvornika med številskimi sistemi za različne vnose podatkov.

Napotki

Pri tej nalogi vam bo koristilo poznavanje osnovnih operacij za delo z nizi in znaki. Dolžino niza pridobimo s pomočjo metode (ne atributa!) `length`, posamezne znake niza pa s pomočjo metode `charAt`.

Znaki so interno predstavljeni s celimi števili (kodami ASCII), zato jih lahko med seboj primerjamo in z njimi tudi računamo. Razlika dveh znakov, denimo, je enaka razliki njunih kod ASCII. Ker so kode ASCII za znake od '0' do '9' ter za znake od 'A' do 'Z' zaporedne (od 48 do 57 za znake od '0' do '9' in od 65 do 90 za znake od 'A' do 'Z'), lahko znak med '0' in '9' pretvorimo v število med 0 in 9 s pomočjo izraza (`znak - '0'`). Število med 0 in 9 pa lahko v znak med '0' in '9' pretvorimo s pomočjo izraza (`char`) (`število + '0'`). Podobno lahko obravnavamo znake med 'A' in 'F'.

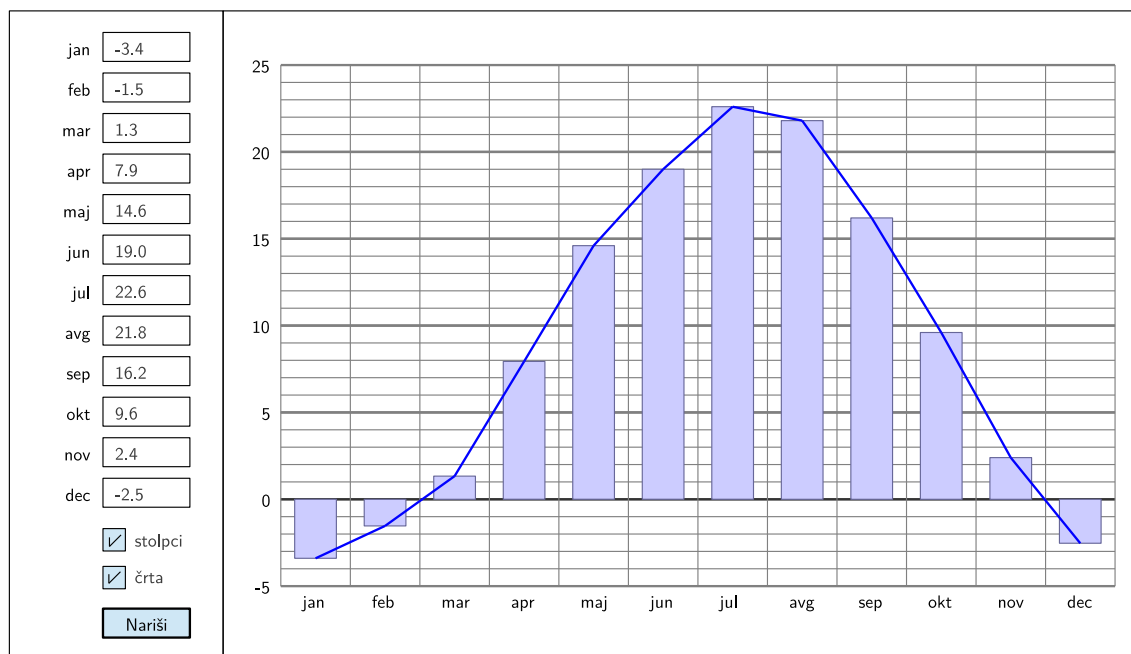
12.2 Diagram temperatur

Naloga

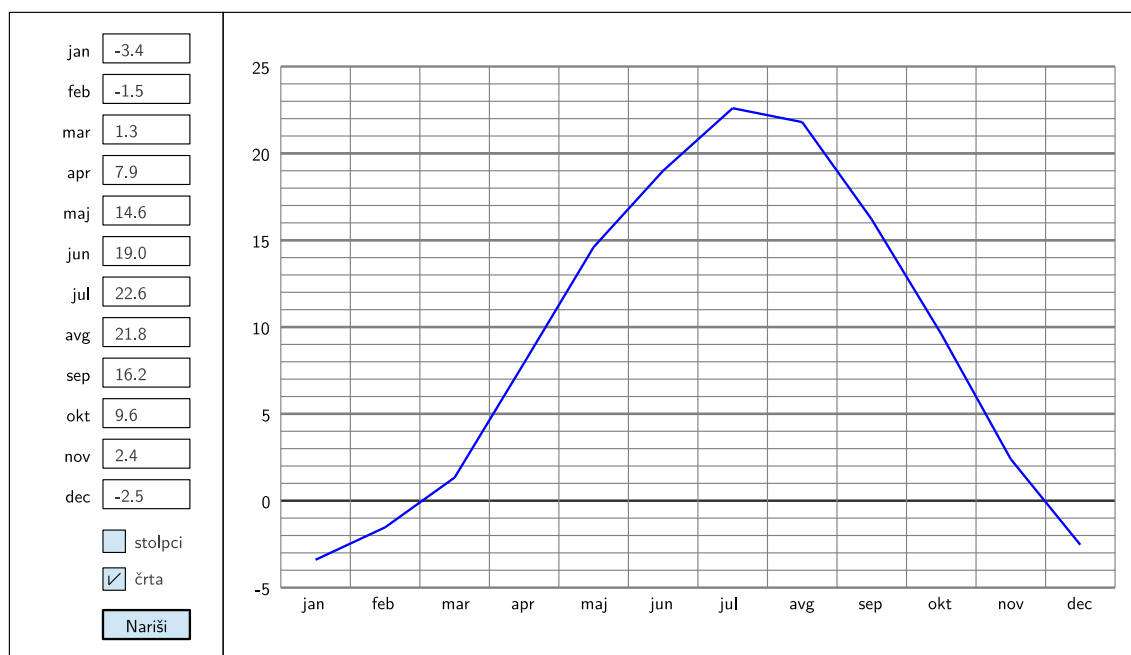
Napišite program za prikaz stolpičnega in/ali črtnega diagrama povprečnih mesečnih temperatur. Program naj se izvaja v oknu, ki vsebuje vnosna polja za vnos posameznih temperatur, potrditveni polji (`JCheckBox`) za določitev načina prikaza diagrama (stolpični diagram, črtni diagram ali oboje), gumb za sprožitev oz. osvežitev izrisa in ploščo za prikaz diagrama. Program naj smiselno deluje tudi v primeru, ko uporabnik ne vnese vseh podatkov.

Primeri

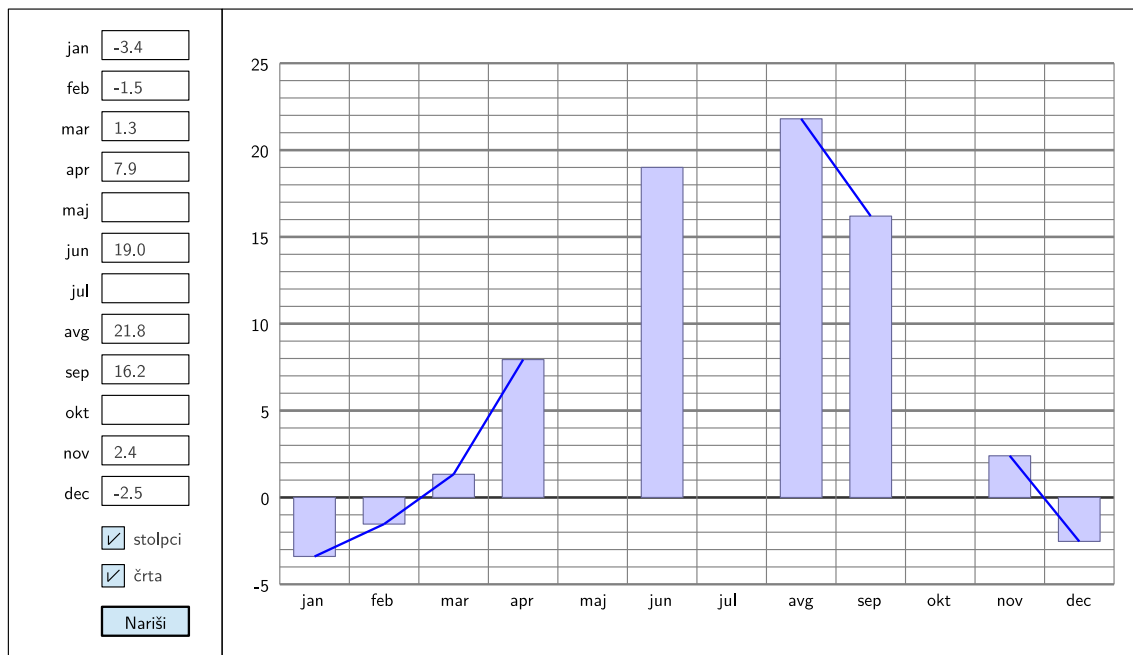
Na slikah 12.2–12.4 so prikazani trije primeri izvajanja programa po vnosu podatkov in pritisku na gumb za izris.



Slika 12.2: Prikaz gibanja temperatur s stolpičnim in črtnim diagramom.



Slika 12.3: Prikaz gibanja temperatur samo s črtnim diagramom.



Slika 12.4: Prikaz gibanja temperatur s stolpičnim in črtnim diagramom pri manjkajočih podatkih.

12.3 Minolovec (★)

Naloga

Napišite program za igro Minolovec. Igra se odvija na pravokotni površini z $A \times B$ polji ($A \geq 4$, $B \geq 4$), pod katerimi je naključno razporejenih M min ($2 \leq M \leq AB - 10$); števila A , B in M naj uporabnik poda kot parametre ob zagonu programa. Naloga uporabnika je določiti položaje vseh M min.

Na začetku igre so vsa polja *zaprta*; uporabnik ne ve, kaj se skriva pod njimi. Nato uporabnik z levim miškinim gumbom *odpira* posamezna polja, dokler ne odpre miniranega polja (v tem primeru takoj izgubi) ali pa dokler ne odpre vseh neminiranih polj (v tem primeru zmaga). Če uporabnik odpre neminirano polje, naj program na tem polju prikaže številko, ki pove, koliko min se nahaja v neposredni okolici (3×3) pravkar odprtega polja. (Največje možno število min v neposredni okolici znaša 8.) Če se v neposredni okolici ne nahaja nobena mina, naj program po enakem pravilu (torej rekurzivno) odpre vsa sosednja polja.

Uporabnik lahko mine tudi aktivno označuje — s pomočjo *zastavic*. Zastavice naj bo možno postavljati in odstranjevati z desnim miškinim gumbom: klik na prazno polje postavi zastavico, klik na polje z zastavico pa zastavico odstrani. Zastavice ne vplivajo na cilj igre (odpreti vsa neminirana polja), ampak služijo zgolj kot pomoč uporabniku. Če uporabnik izgubi (odpre minirano polje), naj program prikaže položaje vseh še ne označenih min in vseh napačno označenih min.

Če uporabnik z levim gumbom klikne na že odprto polje, na katerem se nahaja neka številka n , in če je uporabnik z zastavicami označil natanko n domnevnih min v neposredni okolici, naj program odpre vsa zaprta polja v okolici tega polja. Če je uporabnik katero od n zastavic postavil napačno, bo igro izgubil, saj bo eno od zaprtih polj v okolici gotovo vsebovalo mino. Ta funkcija sicer ni nujno potrebna za igranje Minolovca, vendar pa igro

znatno pospeši.

Mine naj se razporedijo šele po prvem uporabnikovem kliku z levim gumbom, in sicer tako, da v neposredni okolici polja, na katerega klikne uporabnik, ni nobene mine. (Uporabnik tako na začetku vedno klikne na polje s številko 0, kar pomeni, da se bo poleg izbranega polja odprlo še najmanj vseh 8 sosednjih polj.) Na ta način se velikokrat vzpostavi situacija, ki uporabniku omogoča smiselno nadaljevanje igre.

Primer

- Denimo, da uporabnik izbere igro na plošči 7×7 z 10 minami. Denimo, da najprej klikne na polje (5, 5) (6. vrstica, 6. stolpec). Sedaj mora program naključno razporediti 10 min, upoštevajoč dejstvo, da na polju (5, 5) in v njegovi neposredni okolici — torej na poljih (4, 4), (4, 5), (4, 6), (5, 4), (5, 6), (6, 4), (6, 5) in (6, 6) — ne sme biti nobene mine. Denimo, da program mine razporedi takole:

$$\begin{bmatrix} - & - & * & * & - & - & - \\ * & - & - & - & - & - & - \\ * & - & - & * & * & - & * \\ - & - & * & * & - & - & - \\ - & - & - & - & - & - & - \\ - & - & - & - & - & - & - \\ * & - & - & - & - & - & - \end{bmatrix}$$

- Ker okolica polja (5, 5) ne vsebuje nobene mine, bo program po kliku na to polje rekurzivno odprl še vsa sosednja polja. Dobimo sledečo sliko (recimo, da program številke 0 zaradi preglednosti ne označuje):

				3	2	1
	1	2	2	1		
	1					
	1					
Število neoznačenih min: 10						

- Uporabnik zlahka ugotovi, da morata biti na poljih (3, 2) in (3, 3) mini, zato ju označi z zastavicama (desni klik):

		▶	▶	3	2	1	
	1	2	2	1			
	1						
	1						
Število neoznačenih min: 8							

4. Ker je sedaj okrog enice na polju (4, 1) označena natanko ena mina, lahko uporabnik klikne na to enico, nakar program odpre vsa zaprta polja v njeni okolici (če pri tem naleti na polje, ki v neposredni okolici nima nobene mine, rekurzivno odpre vse njegove sosede):

1	2	▶	▶	3	2	1	
	1	2	2	1			
1	1						
	1						
Število neoznačenih min: 8							

5. Mina mora biti bodisi na polju (2, 0) bodisi (2, 1), zato je na polju (2, 2) gotovo ni:

		3					
1	2	▶	▶	3	2	1	
	1	2	2	1			
1	1						
	1						
Število neoznačenih min: 8							

6. Recimo, da uporabnik tvega in označi polje (2, 1) kot minirano:

		▶	3				
1	2	▶	▶	3	2	1	
	1	2	2	1			
1	1						
	1						
Število neoznačenih min: 7							

7. Sedaj lahko klikne na trojko na polju (2, 2). Program bo odprl vsa zaprta polja v okolici tega polja:

		●	●				
●	3	3	4				
●	⊗	3	●	●		●	
1	2	▶	▶	3	2	1	
	1	2	2	1			
1	1						
●	1						
Ojej!							

Žal se je uporabnik zmotil: na polju (2, 1) mine ni, je pa na polju (2, 3). Ta mina eksplodira ob kliku na trojko na polju (2, 2). Igre je konec!

8. Če bi uporabnik v koraku 6 namesto polja (2, 1) z zastavico označil polje (2, 3), bi po kliku na trojko na polju (2, 2) nastala takšna slika:

	3	3	4				
	3	3	▶				
1	2	▶	▶	3	2	1	
	1	2	2	1			
1	1						
	1						
Število neoznačenih min: 7							

9. Sedaj ni težko ugotoviti, da morajo na poljih (1, 0), (2, 0), (2, 4) in (2, 6) biti mine:

13. Polji (0, 2) in (0, 3) sta očitno minirani:

		▶	▶	1		
▶	3	3	4	3	2	1
▶	3	3	▶	▶	2	▶
1	2	▶	▶	3	2	1
	1	2	2	1		
1	1					
	1					
Število neoznačenih min: 1						

14. Sedaj lahko kliknemo na trojko na polju (1, 1):

1	2	▶	▶	1		
▶	3	3	4	3	2	1
▶	3	3	▶	▶	2	▶
1	2	▶	▶	3	2	1
	1	2	2	1		
1	1					
▶	1					
Čestitke!						

Po odprtju polj (0, 0) in (0, 1) se na vseh zaprtih poljih (takšno je le še polje (6, 0)) nahajajo mine, zato program sam dopolni manjkajoče zastavice (v našem primeru je ena sama) in čestita uporabniku.

Napotek

Do parametrov, ki jih uporabnik poda ob zagonu programa, dostopamo preko tabele `args`, ki jo kot svoj argument prejme metoda `main`. Na primer, če program poženemo kot

```
java Minolovec 10 12 20
```

potem tabela `args` vsebuje nize (ne števila!) 10, 12 in 20.

12.4 Elektronska preglednica (★)

Naloga

Napišite preprosto elektronsko preglednico. Okno programa naj prikazuje dva stolpca po N vnosnih polj ($1 \leq N \leq 26$; število N naj vnese uporabnik kot parameter ob zagonu programa), oznake in gumb »Izračunaj!«. Vnosna polja naj bodo označena z zaporednimi črkami angleške abecede. Vnosna polja v levem stolpcu so namenjena vnosu aritmetičnih izrazov, tista v desnem stolpcu pa prikazu rezultatov po pritisku na gumb »Izračunaj!«. S klici `setEditable(false)` in `setFocusable(false)` onemogočite vnos v vnosna polja v desnem stolpcu.

V vnosna polja v levem stolpcu naj bo mogoče vnašati veljavne aritmetične izraze. Veljaven izraz je eno od sledečega (simboli I , I_1 in I_2 označujejo poljubne veljavne izraze):

- ne-negativno celo število
- oznaka vnosnega polja (A , B , C , ...)
- $I_1 + I_2$
- $I_1 - I_2$
- $I_1 * I_2$
- I_1 / I_2 (celoštevilsko deljenje)
- (I)

Primer veljavnega izraza je $32 * (0 - 3 * (C + D + 15) / 3)$. Rezultat izraza je vedno neko celo število (lahko tudi negativno). Lahko predpostavite, da so podatki in rezultati vedno v mejah od `Integer.MIN_VALUE` do `Integer.MAX_VALUE`.

Po pritisku na gumb »Izračunaj!« naj se izračunajo vrednosti izrazov v vseh vnosnih poljih v levem stolpcu, rezultati pa naj se zapišejo v istoležna vnosna polja v desnem stolpcu. Rezultat praznega vnosnega polja je enak 0. Če izraza v nekem vnosnem polju ni mogoče izračunati, naj program namesto rezultata izpiše sledeče:

- **sintaksa**, če je izraz sintaktično napačen (neujemajoči oklepaji, operatorji brez operandov itd.; izraz -3 , denimo, je sintaktično napačen, saj predpostavljamo, da so vsa števila ne-negativna, zato operatorju $-$ v tem izrazu manjka levi operand);
- **sklic**, če je izraz sintaktično pravilen, vendar pa vsebuje krožni sklic (npr. sklic na vnosno polje p v vnosnem polju p ali pa sklic na vnosno polje q v vnosnem polju p , pri čemer vnosno polje q neposredno ali posredno vsebuje sklic na vnosno polje p) ali pa sklic na vnosno polje, ki vsebuje napako.
- **deljenje z 0**, če je izraz sintaktično pravilen in ne vsebuje krožnih sklicev, vendar pa zahteva deljenje z ničlo.

Primer

Slika 12.5 prikazuje vsebino preglednice za $N = 14$ po vnosu izrazov v vnosna polja v levem stolpcu in pritisku na gumb »Izračunaj!«. Ker vnosno polje I vsebuje sintaktično napako (zaporedna operatorja $+$ in $*$), vsebuje vnosno polje G , ki se sklicuje na I , napako sklica. Vnosna polja F , J in L se druga na drugo krožno sklicujejo, zato vsa tri vsebujejo napako sklica. Napako sklica vsebuje tudi vnosno polje H , ki se sklicuje na neizračunljivo vnosno polje F , in vnosno polje E , ki se sklicuje na vnosno polje A , ki vsebuje deljenje z ničlo. Vnosno polje N vsebuje sklic samo nase. Prazno vnosno polje K ima po definiciji vrednost 0. Vrednosti ostalih vnosnih polj se izračunajo po običajnih matematičnih pravilih.

A	<input type="text" value="3/(D-5*B/2)"/>	<input type="text" value="deljenje z 0"/>
B	<input type="text" value="3*C"/>	<input type="text" value="6"/>
C	<input type="text" value="D/6"/>	<input type="text" value="2"/>
D	<input type="text" value="5*6-3*(2+3*5)+11/3*12"/>	<input type="text" value="15"/>
E	<input type="text" value="A"/>	<input type="text" value="sklic"/>
F	<input type="text" value="J+4"/>	<input type="text" value="sklic"/>
G	<input type="text" value="D+I"/>	<input type="text" value="sklic"/>
H	<input type="text" value="3-F"/>	<input type="text" value="sklic"/>
I	<input type="text" value="((K-5))-3/(2+*5)"/>	<input type="text" value="sintaksa"/>
J	<input type="text" value="3-L"/>	<input type="text" value="sklic"/>
K	<input type="text" value=""/>	<input type="text" value="0"/>
L	<input type="text" value="B+2*F"/>	<input type="text" value="sklic"/>
M	<input type="text" value="((K-5))-(3)*(D-(C-B-D))"/>	<input type="text" value="-107"/>
N	<input type="text" value="N+1"/>	<input type="text" value="sklic"/>
<input type="button" value="Izračunaj!"/>		

Slika 12.5: Primer delovanja elektronske preglednice.