

1. naloga: (20 točk, čas reševanja: 20 minut)

Napiši program `Naloga21.java`, ki na zaslon izpiše vse besede, ki jih dobimo z mešanjem črk niza, ki je podan kot prvi argument programa. Nizi naj bodo urejeni po abecedi, vsak niz naj bo izpisan le enkrat.

Primer 1: vhod: `abc`

izhod: `abc, acb, bac, bca, cab, cba`

Primer 1: vhod: `123`

izhod: `123, 132, 213, 231, 312, 321`

Primer 1: vhod: `abba`

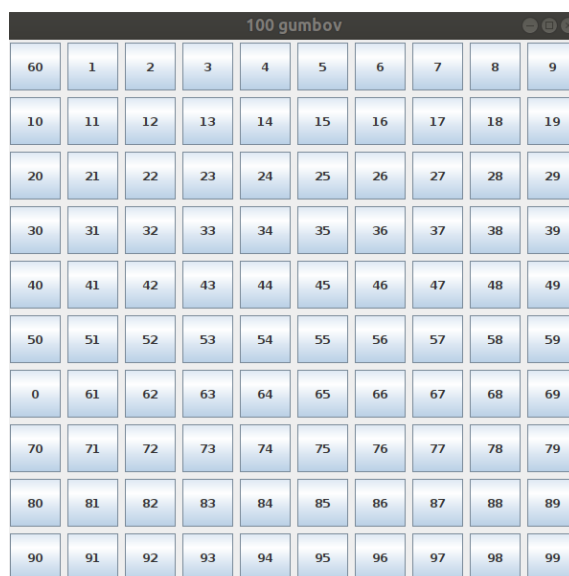
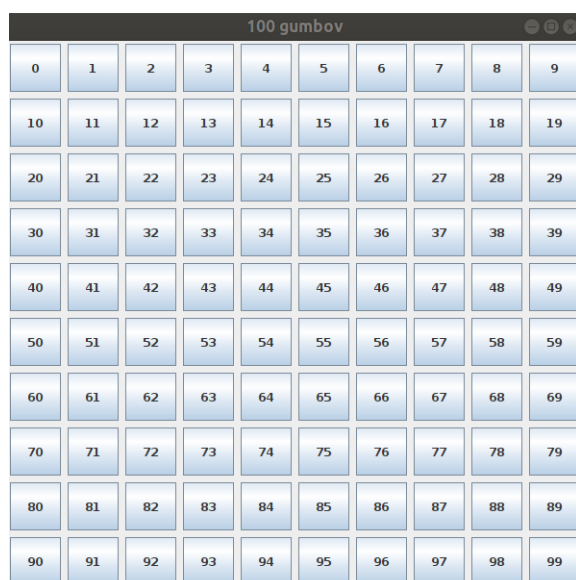
izhod: `aabb, abab, abba, baab, baba, bbaa`

2. naloga (25 točk, čas reševanja: 30 minut)

Napiši program `Naloga22.java`, ki nariše okno in v njem 100 gumbom, kot prikazuje spodnja leva slika. Ob pritisku na katerikoli gumb, naj se zgodi dvoje:

- besedilo na gumbu se izpiše na zaslon (uporabite metodo `println()`),
- zamenja naj se besedilo na pritisnjenem in drugem, naključno izbranem gumbu.

Primer: na desni sliki je prikazano stanje okna po pritisku na gumb 0 (program je na zaslon izpisal "0", izbral naključni gumb (60) in zamenjal napisa "0" in "60" na gumbih).



3. naloga (25 točk, čas reševanja: 30 minut)

Ker je Kekec že odrasel in Bedancu ne povzroča več preglavic, se je le ta lahko na stara leta začel ukvarjati z bolj produktivnimi in družbeno koristnimi stvarmi. Tako se je pridružil inštitutu za gozdarstvo in začel po slovenskih gozdovih spremljati medvede. Da pa bi lahko medvede spremljali tudi takrat, ko ljudje spimo in smo daleč od njih, je skupaj z lovci nekaj medvedom nadel ovratnice opremljene z GPS in GSM enoto, ki neprestano spremlja položaj medveda in podatke preko GSM povezave posreduje Bedancu, ki lahko tako medvede spremlja kar iz pisarne.

Zaradi prihranka prostora in tehnologije medvedjih ovratnic, dobi Bedanec podatke v surovi - binarni obliki, ki je zelo težko berljiva in takšne podatke težko uporabi za nadaljnje raziskave. Vaš cilj je tako pomagati Bedancu in izdelati program, ki bo podatke zapisane v binarni datoteki spremenil v formatirano besedilno datoteko, kjer so posamezni podatki med seboj ločeni s presledki in poravnani po stolpcih.

V vhodni binarni datoteki (primer: `medved.bin`) so za vsak položaj medveda zapisani naslednji podatki:

- leto ... število tipa int
- mesec ... število tipa int
- dan ... število tipa int
- ura ... število tipa int
- minuta ... število tipa int
- latitude ... število tipa float
- longitude ... število tipa float

V izhodni tekstovni datoteki naj bodo podatki o položaju medveda izpisani v eni vrstici. Primer, ob klicu programa

```
java Naloga23 medved.bin medved.txt
```

naj se ustvari tekstovna datoteka `medved.txt`, v njej pa naj bodo podatki izpisani takole:

year	month	day	hour	minute	latitude	longitude
2009	5	23	22	12	46.495098	15.508233
2009	5	23	22	12	46.495216	15.508200
2009	5	23	22	12	46.495251	15.508200
2009	5	23	22	12	46.495285	15.508217
2009	5	23	22	12	46.495449	15.508133
2009	5	23	22	12	46.495617	15.507950
2009	5	23	22	13	46.495716	15.507850

4. naloga (30 točk, čas reševanja: 40 minut)

Napiši program `Naloga24.java`, ki poišče pot po najcenejših in po najdražjih povezavah v grafu.

V datoteki (njeno ime je podano kot prvi argument programa) je podan seznam povezav grafa, kjer je vsaka povezava opisana v eni vrstici z naslednjimi tremi podatki: `začetno_vozlišče_povezave`, `končno_vozlišče_povezave` in `cena_povezave`. Začetno in končno vozlišče sta predstavljena z nizom znakov (`String`), cena povezave pa s celim številom (`int`). Podatki so ločeni s presledkom.

Naloga:

- 1) Napiši razred `Povezava` za hranjenje podatkov o eni povezavi (oznaka začetnega in končnega vozlišča, cena povezave). Atributi razreda naj bodo privatni, razred pa naj vsebuje primeren konstruktor ter metodo `toString()`.

- 2) Napiši metodo

```
ArrayList<Povezava> preberiPovezave(String imeDatoteke),
```

ki prebere vsebino podane datoteke in vrne seznam prebranih povezav.

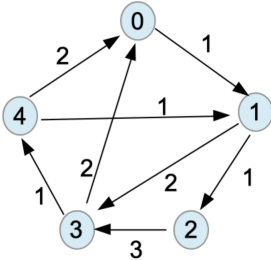
- 3) Napiši metodo

```
ArrayList<String> poisciPot(  
    ArrayList<Povezava> povezave, String zacetek, int kako),
```

ki poišče in vrne pot po najcenejših (`kako==0`) oziroma najdražjih (`kako==1`) povezavah. Pot naj se začne v točki `zacetek` in konča, ko prvič pripelje v eno od že obiskanih točk.

Program naj na koncu izpiše zaporedje obiskanih vozlišč.

Primer:

graf.txt	Pripadajoči graf:	Izpis ob klicu <code>java Naloga24 graf.txt 0</code>
0 1 1 1 2 1 1 3 2 2 3 3 3 0 2 3 4 1 4 1 1 4 0 2		Pot po najcenejših povezavah: [0, 1, 2, 3, 4, 1] Pot po najdražjih povezavah: [0, 1, 3, 0]

Opomba: Predpostaviš lahko, da so podatki v vhodni datoteki vedno pravilni ter da v grafu obstaja podano začetno vozlišče.