Gyakorló feladatok

Az alaputasítások olyanok, mint C-ben. (Részleteket ld. segedletek/02.pdf vagy bármelyik Java tutorial.) Ha úgy érzi, hogy nagyon hosszú volt a nyár, és igencsak elfelejtette az alapokat, oldjon meg néhányat ezek közül a feladatok közül. Ha úgy gondolja, biztos alapokkal rendelkezik, akkor sem árt legalább fejben végiggondolni, hogy hogyan is lehetne megoldani őket.

1. Adott egy pozitív egész számokból álló tömb (egyelőre megadhatja konstansként). Hozzon létre egy másik tömböt, amely az előzőben lévő számok faktoriálisát tartalmazza. A megoldáshoz használjon metódusokat.

2. A megadott tömb elemei közül írassa ki
   a/ a páratlan számokat
   b/ a k-val osztható számokat (k-t most megadhatja a program elején konstansként)
   c/ a prímszámokat

Segítség:
Egy szám prím, ha 1-en és önmagán kívül nincs más osztója ☺
A ciklusváltozót 2-ről indítva meg kell nézni, hogy a ciklusváltozó osztója-e a vizsgált számnak. Ha igen, akkor az illető szám nem prím.

Ha a szám gyökéig nincs osztója, akkor utána sincs - vagyis elég csak a gyökig vizsgálni az osztókat. (Gyök: double Math.sqrt(double x))

Úgy kellene megoldani, hogy ha egyáltalán nem talál prímszámot, akkor írja ki, hogy a beolvasott számok között nincs prím.

3. Írja meg a faktoriális-számítást rekurzív módon!

   A faktoriális rekurzív definíciója: 1!=1; n!=n*(n-1)!

   (További ötlet: fakt(n) = n*fakt(n-1).)

4. Most kiindulásként 1 és n közötti véletlen pozitív egész számokból álló tömböt vegyen. (n-t megadhatja konstansként a program elején).

Segítség:
A double értékű 0 <= Math.random() <1 metódus egy [0,1) közötti véletlen értéket állít elő. Gondolja végig, hogy ebből hogyan lehet 0 és n közötti egész értéket képezni.

5. Módosítsa valamelyik előző feladatot úgy, hogy olvassa be a szükséges adatokat.
A feladat megoldása során figyeljen arra is, hogy a megadott számok csak pozitív egész értékek lehetnek. (Nem biztos, hogy még mindenre tud figyelni, de negatív egész esetén adjon hibaüzenetet.)

Segítség:
Adatok beolvasásának több módja is lehet, ezek közül egy szerepel a gyak1.pdf fájlban.

6. a/ Készítsen kódtáblázatot! Írassa ki 1-től 255-ig az ASCII kódokat, és mellé a kódnak megfelelő karaktert.
b/ Gyűjtse egy tömbbe az angol ábécé betűit.

Segítség:
A kód maga a szám, ha ezt a számot karakterként (típuskényszerítés) íratjuk ki, akkor a képernyőn a kódnak megfelelő karakter jelenik meg.
A 255-t NE DRÓTOZZA BE FIXEN a ciklus fejébe, hanem deklarálja konstansként. Konstans megadásakor a változódeklaráció elé a final kulcsszó kerül.
Az angol ábécé betűinek kódja 65-től 90-ig és 97-től 122-ig terjed, de ezeket se drótozza be fixen, hanem itt is használja a final módosítót.

7. a/ Készítsen Java programot a féléves jegy kiszámítására. Beolvasandó adatok: a két gyakorlati zh eredménye (százalékban). Ha ezek átlaga eléri a kívánt határt, akkor már csak az elméleti zh eredményét kell beolvasni, ha nem, akkor a javító zh-ét is. Figyelje azt is, hogy egyáltalán eljuthat-e az illető az elméleti zh-ig.
A jegyszámításhoz szükséges határokat NE FIXEN adja meg, viszont megadhatja őket konstansként a program elején.

b/ Készítsen féléves-jegy „kalkulátort”. Vagyis, ha eddig nem tette volna, akkor most alakítsa át az előző feladatot, és írjon egy metódust egyetlen ember jegyének meghatározására, és ezt a metódust felhasználva osztályozza az embereket, amíg van érdeklődő ember (nem tudjuk előre az emberek számát). Az illetőtől kérje be a nevét, majd a metódus lefutása után írja mellé az osztályzatát, majd kérje be a következő ember nevét, ha van még következő ember (ezt egy kérdésre adott válasszal tudja eldönteni).

8. a/ Adott egy évszám, döntse el, hogy szökőév-e. A szökőév eldöntéséhez írjon egy boolean metódust.
b/ Írassa ki adott (beolvasott) két évszám közötti szökőéveket, és gyűjtse is őket egy tömbbe!

Szökőévek a következők: minden néggyel osztható év, kivéve a százzal is oszthatókat. Szökőévek viszont a 400-zal osztható évek. Vagyis a százasra végződő évek közül csak azok szökőévek, amelyek 400-zal is oszthatók.
Ez alapján tehát szökőév volt 1988, 1992, 1996. Nem szökőév 1700, 1800, 1900, 2100, és 2200. Viszont szökőévek az alábbi esztendők: 1600, 2000 és 2400.

9. a/ Generálja az n és m közötti háromjegyű palindromokat! n, m konstansként megadható a program elején ($100 \leq n \leq m \leq 999$). Palindrom az olyan háromjegyű szám, amelynek első és utolsó számjegye azonos. Ha átnézte a beolvasást, akkor olvassa be n és m értékét, és természetesen végezze el a szükséges feltételvizsgálatokat.

b/ Állapítsa meg egy billentyűzetről beolvasott 100 és 999 közé eső háromjegyű számról, hogy palindróma-e! Azt is ellenőrizze, hogy tényleg háromjegyű számról van-e szó.

c/ Adjon rá lehetőséget, hogy ne csak egy számról lehessen eldönteni a fenti kérdést, vagyis az első szám vizsgálata után kérdezze meg, hogy a felhasználó akar-e még másik számot is vizsgáltatni, és a választól függően vagy kérje be az újabb számot, vagy köszönjön el a felhasználótól.

10. A következő feladatot metódusok használatával oldja meg! A metódusok a következők:

a) Véletlenszerűen töltsön fel egy legföljebb 100 elemű tömböt 'a' és 'z' közötti betűkkel. (Ugye eszébe sem jut fixen bedrótozni a 100-at? A metódust tetszőleges elemszámú tömbre írja!)

b) Keresse meg a tömb legnagyobb elemét (az ABC-ben legkésőbb szereplő betű) (a main metódus írja ki a betűt, vagyis ez a metódus egy karaktert ad vissza).

c) Írja ki, hogy egy billentyűzetről bekért betű hányszor szerepel a tömbben (a main metódus végzi a betű bekérését és a darabszám kiírását, ez a metódus a darabszámot adja vissza a beolvasott betű függvényében).

d) Írja ki, hogy hány darab magánhangzó szerepel a tömbben.

e) Készítsen egy mátrixot, amely [betű]-[betű előfordulásának száma] kettőst tartalmaz. (a main metódus írja ki a mátrixot).

11. Egy börtönben 1000 cella van, mindegyikben egy rab ül. Kezdetben minden cella zárva van. A börtönőrnek játszani támad kedve: végigmegy az összes cella előtt, és mindegyik ajtó zárján fordít egyet. Fordításkor a nyitott cellát bezárja, illetve a zártat kinyitja. Ha végigment, elkezdi elölről, és minden második cella zárján fordít egyet. Aztán minden harmadikon fordít, és így tovább. Legvégül fordít egyet az ezrediken, és kész.
Ezután amelyik cella ajtaja nincs bezárva, abból a rab elmehet. Kik a szerencsés rabok?

12. a/ Töltsön fel oszlopfolytonosan 1-12-ig terjedő számokkal egy 3x4-es mátrixot. A kívánt végeredmény:

1 4 7 10
2 5 8 11
3 6 9 12

b/ Oldja meg az előző feladatot n × m -es mátrixra! (És szokja meg, hogy SOHA NE dolgozzon fix adatokkal – az előbbi feladatban csak azért kértük, hogy könnyebben átlássa a mátrix indexelését.)
Számolja ki a mátrix főátlójában lévő elemeinek összegét, feltéve, ha van főátló! Ha nincs, jelezze! (Főátló: bal fölső sarokból a jobb alsó sarokba vezető átló.)

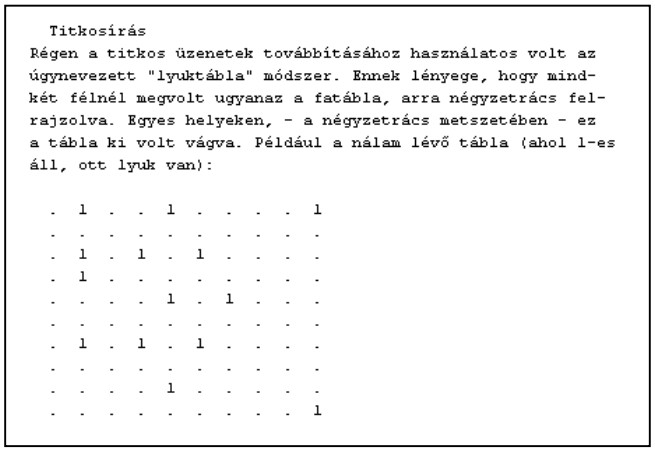
Segítség:
Java-ban nincs kétdimenziós tömb (mátrix), csak tömbök tömbje.

Pl.:

```
int sor = 3;
int oszlop = 5;
float matrix [ ][ ] = new float[sor][oszlop];
```

(utolsó elem: matrix[2][4])

13. Készítsen titkosíráshoz használható sifrét. (Aki olvasta, Jules Verne Sándor Mátyás c. könyvében leveleztek ilyesmivel.) Ehhez a következőre van szükség.
- Egy n × m -es, karakterekkel feltöltött mátrix. (A karakterek közé van elrejtve maga az üzenet is.)
- Egy szintén n × m -es mátrix, ez lesz a „sifre”, vagyis 1-es, ahol azt szeretnénk, hogy látszódjon a karakter, 0, ahol nem.
- mátrix: a titkosítás eredménye: kimaszkolni az eredeti mátrixot úgy, hogy csak az 1-esek helyén lévő betű látszódjon - ekkor kiolvasható a titkos szöveg.

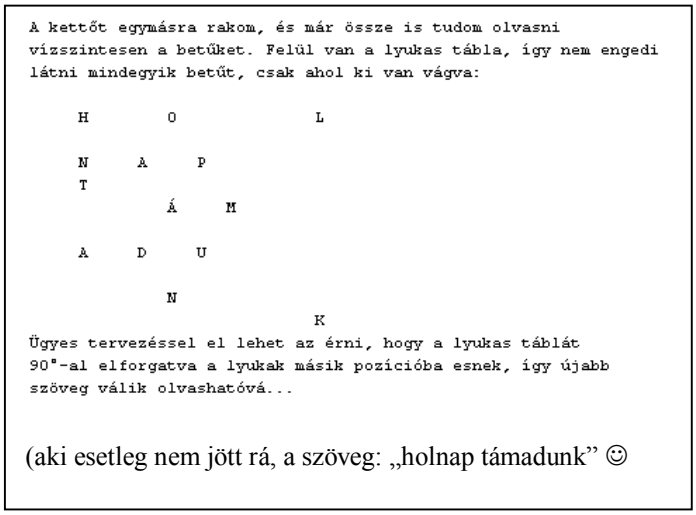


Titkosírás
Régen a titkos üzenetek továbbításához használatos volt az úgynevezett "lyuktábla" módszer. Ennek lényege, hogy mindkét félnél megvolt ugyanaz a fatábla, arra négyzetrács felrajzolva. Egyes helyeken, - a négyzetrács metszetében - ez a tábla ki volt vágva. Például a nálam lévő tábla (ahol 1-es áll, ott lyuk van):

```
. 1 . . 1 . . . . 1
. . . . . . . . . .
. 1 . 1 . 1 . . . .
. 1 . . . . . . . .
. . . . 1 . 1 . . .
. . . . . . . . . .
. 1 . 1 . 1 . . . .
. . . . . . . . . .
. . . . 1 . . . . .
. . . . . . . . . 1
```

Ezt a táblát hozta a futár:

```
F H L T O D V S L L
F D G E H C X W F C
W N F A B P N L Z N
U T S K Q T J O R Z
F K Q B Á K M U Y Q
B B B U V L F M C B
S A K D D U J L A X
W J C A K W O Q N R
M V D D N Z S Q S U
R G W B C B U R S K
```



A kettőt egymásra rakom, és már össze is tudom olvasni vízszintesen a betűket. Felül van a lyukas tábla, így nem engedi látni mindegyik betűt, csak ahol ki van vágva:

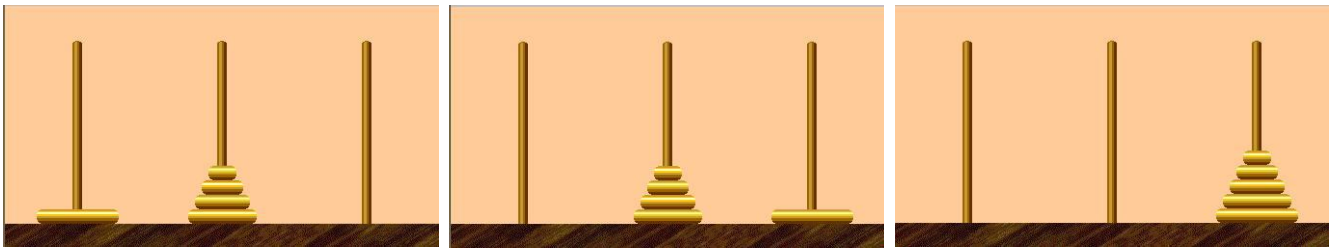
```
     H        O             L

     N     A     P
     T
               Á     M

     A     D     U

               N
                             K
```

Ügyes tervezéssel el lehet az érni, hogy a lyukas táblát 90°-al elforgatva a lyukak másik pozícióba esnek, így újabb szöveg válik olvashatóvá...

(aki esetleg nem jött rá, a szöveg: „holnap támadunk” ☺

14. Írjon programot a Hanoi torony problémájának megoldására! A probléma:
Adott három rúd: A, B és C. A rúdon induláskor N darab különböző átmérőjű lyukas korong helyezkedik el egymás fölött csökkenő sorrendben.
Feladat: a korongok áthelyezése az alábbi szabályok alapján:

– a B rúd közbenső tárolásra használható
– minden lépésben csak egy korong mozgatható
– minden korong csak nála nagyobb átmérőjű korongra helyezhető.



Természetesen nem feladat még a grafikus felület, most csak az alábbi formában kérjük az eredményt:

```
--------------------Configuration: <Default>--
Hány koronggal dolgozzam?: 4
rakj egy korongot bal rúd --> középső rúd
rakj egy korongot bal rúd --> középső rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot bal rúd --> középső rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot középső rúd --> jobb rúd
rakj egy korongot bal rúd --> középső rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot középső rúd --> jobb rúd
rakj egy korongot jobb rúd --> bal rúd
rakj egy korongot középső rúd --> jobb rúd
rakj egy korongot középső rúd --> jobb rúd
rakj egy korongot bal rúd --> középső rúd
```

Segítség: célszerű rekurziót alkalmazni.