Programming 1 – Exam 2
February 10, 2021

Submit the files `Prva.java`, `Druga.java`, `Tretja.java`, and `Cetrta.java`. You can test them as follows:

(1) `tj.exe Prva.java . .` (2) `tj.exe Druga.java . .` (3) `tj.exe` (4) `tj.exe`

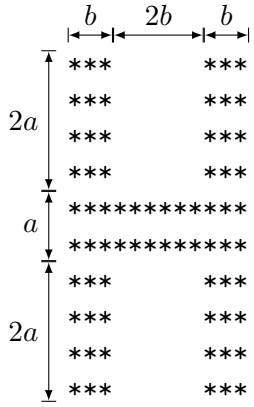
(1) Write a program that reads integers $a \in [1, 50]$ and $b \in [1, 50]$ and, using asterisks and spaces, produces an output as illustrated by the following example. Your program should not print any superfluous space.

In 50% of the hidden test cases, it holds that $a = b$.

Here is a sample input, the corresponding output, and an explanation:

```
2 3
```

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



(2) A mountain chain is represented by a binary matrix in which zeros represent the sky and ones represent the earth. Every column of the matrix consists of (a possibly empty) sequence of zeros and a nonempty sequence of ones.

A mountaineer starts his hike on the first summit and completes it on the last. Write a program that prints the total length of his ascent and the total length of his descent.

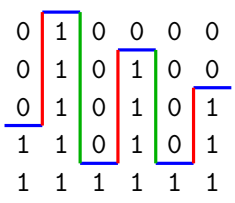
The first line contains integers $h \in [1, 100]$ and $w \in [1, 100]$. This line is followed by the description of the matrix — h lines, each having w zeros and ones. Consecutive numbers in the same line are separated by a single space. Your program should print two lines: the first should contain the length of the ascent, and the second should contain the length of the descent.

In 50% of the hidden test cases, the total length of the descent is equal to 0.

Here is a sample input, the corresponding output, and an explanation:

```
5 6
0 1 0 0 0 0
0 1 0 1 0 0
0 1 0 1 0 1
1 1 0 1 0 1
1 1 1 1 1 1
```

```
8
7
```



Ascent: $3 + 3 + 2 = 8$
Descent: $4 + 3 = 7$

(3) You are given the following classes (as static inner classes in the class `Tretja`):

```
class Oseba {      // a tourist (person)
    private String ip;       // name and surname, e.g., Ana Arko
    private String drzava;   // country of residence, e.g., Slovenia
}
class Cilj {       // a tourist destination
    private String kraj;     // place, e.g., Vienna
    private String drzava;   // country, e.g., Austria
}
class Nocitev {    // an overnight stay by some person at some destination
    private Oseba oseba;     // the person
    private Cilj cilj;       // the destination
    private int leto;        // the year in which the stay took place
}
```

Complete the outer class (`Tretja`) with the following methods:

- [34%] `public static int notranje(Nocitev[] nocitve)`

  Returns the number of overnight stays from the array `nocitve` in which the tourist stayed in his/her own country of residence.

- [32%] `public static boolean jeZvesta(Nocitev[] nocitve, Oseba oseba)`

  Returns `true` if and only if, according to the data in the array `nocitve`, the given tourist (`oseba`) always stayed at the same destination (and never anywhere else, not even within the same country). If the person has never done any overnight stay, the correct answer is `true`.

- [34%] `public static int[][] obiskanost(Nocitev[] nocitve, Cilj[] cilji, int minLeto, int maxLeto)`

  Creates and returns an array of size $C \times L$, where C is the length of the array `cilji` and L is the length of the closed interval [`minLeto`, `maxLeto`]. The element at the row index `i` and column index `j` in the returned array should specify the number of overnight stays from the array `nocitve` that took place at the destination `cilji[i]` in the year `minLeto + j`.

(4) Write the following methods:

- [60%] `public static <T> List<T> razmnozi(List<T> seznam, int n)`

  Creates and returns a list composed of one copy of the first element of the given list, two copies of the second element, ..., n copies of the n-th element, one copy of the $(n+1)$-st element, two copies of the $(n+2)$-nd element, ..., n copies of the $(2n)$-th element, one copy of the $(2n+1)$-st element, etc.

- [40%] `public static <T> Iterator<T> razmnozevalnik(List<T> seznam, int n)`

  Returns an iterator that visits the first element of the given list once, the second element twice, ..., the n-th element n times, the $(n+1)$-st element once, the $(n+2)$-nd element twice, etc.

  The iterator has to walk the list `seznam`. The method must not create a "multiplied" copy of the list and return an iterator that trivially walks the copy!