ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Spring 2013

PROGRAMMING LANGUAGES I

Exercise 2

Deadline of electronic submission: 6/6/2013, 23:59:59

Tassos and Golfo... (0.25+0.25 = 0.5 points)

You are given a map consisting of **N×N** small squares. Each square may be empty (denoted by a space) or contain an obstacle (denoted by the letter "X"). In addition, in one square we find Tassos (denoted by the letter "T") and in another one Golfo (denoted by the letter "G").

Tassos wants to meet with Golfo. Each time, he can move in one of the neighboring squares (up, down, left, and right). Left or right moves cost 1 unit, up moves cost 2 units, while down moves (also known as «falls») cost nothing (other than in hospital costs).

What this exercise asks you to do is to write two programs (one in ML and one in Java) that take the map as input and return as output an integer: the minimum cost for Tassos to reach Golfo. If it is not possible for Tassos to reach Golfo the programs should return -1 (~1 in SML).

The input is to be read from a file, as shown in the examples that follow. The first line contains a natural number **N**: the map's dimension. The remaining **N** lines contain the map. Each line consists of **N** characters where each of them is either a space or one of the letters "X", "T" or "G". There is exactly one "T" and exactly one "G" on the map. You can also take for granted that all characters at the edges of the map are always "X".

Constraints: 4 ≤ **N** ≤ 500, execution time limit: 10 seconds, memory limit:: 256 MB.

Below we show some example calls to the two programs in SML and in Java.

In SML/NJ

```
- golfo "golfo1.txt";
val it = 20 : int
- golfo "golfo2.txt";
val it = ~1 : int
```

In Java

```
> java Golfo golfo1.txt
20
> java Golfo golfo2.txt
-1
```

Where the two files with the maps are the following ones (**`cat`** is a command of Unix):

```
> cat golfo1.txt
10
XXXXXXXXXX
X        X
X T      X
XXXXXXXX X
X      X X
X  X     X
X  XXXXXXX
X       GX
X        X
XXXXXXXXXX
```

```
> cat golfo2.txt
8
XXXXXXXX
X T    X
XXXXXX X
X  X   X
X  XXXXX
X      X
X     GX
XXXXXXXX
```

Tassos and the sheep... (0.25+0.25 = 0.5 points)

Unfortunately, due to the crisis, Tassos cannot just spend all his day with Golfo, but also needs to work from time to time in order to pay his taxes. His job is to "arrange" sheep in the sheepfolds of his village: each morning he gets in his house a list with all the sheep that need to be moved from one sheepfold to another and wants to complete the task with the minimum effort possible. He moves on foot and carries sheep on his shoulders, one on top of the other if necessary. Thus, when he wants to unload a particular sheep in some sheepfold, he first has to unload all other sheep that are on top of that sheep on his shoulders. (In other words, his shoulders function like a stack.) The actions that Tassos can do are:

- **`move k:`** walk from the point that he is at to sheepfold **`k`**, with cost 3 units
- **`load p:`** load the sheep **`p`** from the sheepfold where both Tassos and the sheep are at the time onto his shoulders, on top of all other sheep that may exist there, with cost 1 unit
- **`unload p:`** unload the sheep **`p`** that is on top of his shoulders in the sheepfold he is at at that time, with cost 1 unit
- **`flute:`** do nothing of the above and just... play his flute, with cost 0 units :-)

What this exercise asks you to do is to write two programs (one in ML and one in Java) that take as input the list of sheep that need to be moved between sheepfolds and return as output an integer: the minimum effort cost that Tassos needs to move all sheep to their proper sheepfolds. Assume that initially Tassos is at his home with nothing on his shoulders and needs to move to a sheepfold if he wants to pick a sheep from there.

The input is to be read from a file, as shown in the examples that follow. The first line contains two natural numbers **K** and **N**: the number of sheepfolds and the number of sheep that need to be moved, respectively. The next **N** lines contain information about the moves. Each one consists of two integers **S** and **D**: the sheepfold in which the sheep is originally and the sheepfold that needs to be moved to. You can assume that **S** and **D** are different between them.

Constraints: 0 ≤ **K** ≤ 42, 0 ≤ **N** ≤ 42, memory limit: 4 GB, execution time limit: 2 hours.

Below we show some example calls to the two programs in ML and in Java.

In SML/NJ

```
- provata "provata1.txt";
val it = 8 : int
- provata "provata2.txt";
val it = 13 : int
- provata "provata3.txt";
val it = 20 : int
```

In Java

```
> java Provata provata1.txt
8
> java Provata provata2.txt
13
> java Provata provata3.txt
20
```

Where the three files with information about the moves of the sheep are as follows:

```
> cat provata1.txt
2 1
1 2
```

```
> cat provata2.txt
3 2
1 2
1 3

> cat provata3.txt
3 4
1 2
1 3
2 1
3 2
```

For the first input file the solution is obvious: Tassos needs to move from his house to sheepfold 1, load the sheep on his back, move to sheepfold 2 and unload it there with total cost 3+1+3+1=8. In between, he can play his flute as many times as he wishes, but this does not change the cost.

Without music breaks, the second input file has two minimum cost solutions, both with cost: 3+1+1+3+1+3+1=13.

Finally, for the third input file, Tassos can choose between a solution that starts from sheepfold 1 having cost 3+1+1+3+1+1+3+1+1+1+3+1=20 and one that starts from sheepfold 2 having cost 3+1+3+1+1+1+3+1+1+3+1+1=20. There are also sequences of actions with bigger cost (e.g. one which starts from sheepfold 3 with a total cost of 23) but these are not what the exercise asks for.

More information on the exercises

- You can work in pairs (form teams of two students) if you want to. You can form a different team than the one of the previous exercise set; for this exercise set the teams will need to be recreated anyway on the submission site.
- It's not permitted to share your programs with other students or post them in places that they can be found (e.g. on the internet, in blogs, discussion fora, etc.). In case of considerable similarities between programs of separate teams, both teams will receive zero points, independently of which team got inspired from the other. On the other hand, it is permitted to use auxiliary code from the internet (e.g. the implementation of a sorting algorithm, some data structure implementation, etc.) provided you clearly state in comments the source of each piece of code you use.
- The code of programs in ML must be in one file and work with SML/NJ v110.74, or MLton 20100608, or Objective Caml version 3.11.2. The submission system allows you to select between these three ML dialects.
- The code of programs in Java can be in more than one file but the code of these files should be compilable without problems with commands of the form **`javac Golfo.java`** or **`javac Provata.java`**.
- You can submit your programs using your moodle account (**http://moodle.softlab.ntua.gr**) using the site **http://eager.softlab.ntua.gr/plgrader**. Your programs should read the input as shown in the examples and should not contain any other output because they will not be accepted by the submission site.