Suprnation
Open Book Coding Exercise

Thank you for applying for a position with the SuprNation development team. This exercise is intended to help you understand your ability to undertake research whilst delivering a solution under time pressure. This is an open-book test; you will not be penalised for using appropriate references and external libraries and this is in fact encouraged.

YOU WILL BE ASSESSED ON TWO GROUNDS; **fulfilling the task at hand** and **elegance of the coded solution**. Good luck with the exercise, and we look forward to seeing your code!

Submission Guidelines

We are firm believers in version control system at SuprNation. We would also like to see how your solution evolves. As such, we would like you to use Git to manage your version history. Please ensure that you make regular commits of your code with a descriptive message to ensure that we can understand how your solution evolved as you tackled the requirements. [1]

The top level of your code-base should contain a file `README.md`, which should explain how to run your code, and a file `ASSUMPTIONS.md`, which details any assumptions you made during implementation, the reasons you made these assumptions and why they were justified.

Please upload your code to Github or Bitbucket and send the project url to `mark@suprnation.com`.

[1] You may use any language you consider reasonable to tackle the problem at hand in the given time and the given evaluation criteria described above. At a technical level, you are not being marked on a particular language, but rather on your ability to harness some language to solve the problem at hand.

Minimum/Maximum Triangle Paths

Consider the following triangle of numbers
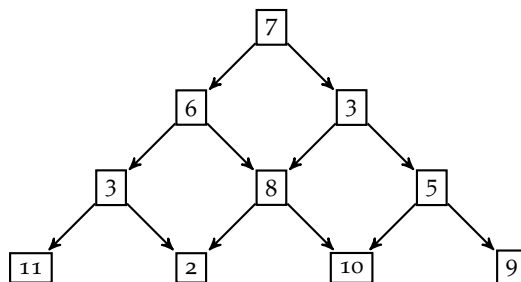
A **path** through the triangle is a sequence of adjacent nodes, one from each row, starting from the top. So, for instance, $7 \rightarrow 6 \rightarrow 3 \rightarrow 11$ is a path down the left hand edge of the triangle.

A **minimal path** is defined as one whose sum of values in its nodes is no greater than for any other path through the triangle. In this case, $7 + 6 + 3 + 2 = 18$ is the minimal path. [2]

We can store the triangle in a text file with each row on a separate line, and spaces between the numbers. [3]

Task

Write a **command-line** program that reads a **text-format triangle** from **standard input** and outputs a **minimal path** to the **standard-output** as follows: [4]

```
$ cat << EOF | java MinTrianglePath
> 7
> 6 3
> 3 8 5
> 11 2 10 9
> EOF
Minimal path is: 7 + 6 + 3 + 2 = 18
```

[2] You can assume that there will be only one minimal path

[3] Thus the triangle above can be stored in text format as:
```
 7
 6 3
 3 8 5
 11 2 10 9
```

[4] Besides producing correct answers your code should

1. use functional concepts - i.e. think higher order functions

2. be capable of producing the correct answer for 500-row triangle