Downloads

- Executable JAR of the visualizer
- Source code of the visualizer

In order to use the offline tester / visualizer tool for testing your solution locally, you'll have to modify your solution by adding the main method that interacts with the tester / visualizer via reading data from standard input and writing data to standard output. As long as you do not change the implementation of method *minimumWork*, this doesn't affect the way your solution works when being submitted to our server.

To simulate a single test case, your program should implement the following protocol (each integer/real number is to be read from / printed in a separate line):

- Read integer *N*.
- Read real numbers **x**[0], **x**[1], ..., **x**[*N*-1].
- Read real numbers **y**[0], **y**[1], ..., **y**[*N*-1].
- Read real numbers **r**[0], **r**[1], ..., **r**[*N*-1].
- Read real numbers **m**[0], **m**[1], ..., **m**[*N*-1].
- Call *minimumWork*(**x**, **y**, **r**, **m**). Let *ret* be the return value.
- Print real numbers *ret*[0], *ret*[1], ..., *ret*[*N*-1]. Flush standard output stream.

In other words, you should implement the following pseudocode in the main method of your solution:

```
N = parseInt(readLine())
for (i=0; i < N; i++)
    x[i] = parseDouble(readLine())
for (i=0; i < N; i++)
    y[i] = parseDouble(readLine())
for (i=0; i < N; i++)
    r[i] = parseDouble(readLine())
for (i=0; i < N; i++)
    m[i] = parseDouble(readLine())

ret = minimumWork(x, y, r, m)

for (i=0; i < 2*N; i++)
    printLine(ret[i])

flush(stdout)
```

In order to run the tester / visualizer, you should use the following command:

```
java -jar CirclesSeparationVis.jar -exec "<command>"
```

`<command>` is the command you would use to execute your solution. If your compiled solution is an executable file, the command will just be the full path to it, for example, "C:\TopCoder\solution.exe" or "~/topcoder/solution". In case your compiled solution is to be run with the help of an interpreter, for example, if you program in Java, the command will be something like "java -cp C:\TopCoder Solution".

Additionally you can use the following parameters (all are optional):

- `-seed <seed>`. Sets the seed used for test case generation. Default value is 1.
- `-novis`. Switches the visualization off, leaving only text output.
- `-sz <window size>`. Sets the size of visualizer window, in pixels. The default value is 700.

The visualizer draws the final location of each circle. Colors represent masses of circles: darker circles are heavier than lighter ones. The red lines show the trajectories along which each circle was moved. You can use space key to toggle between with/without trajectories modes. The green square is the 0 <= x, y <= 1 square from which original circles' centers were chosen.

You can print any debug information of your solution to the standard error stream and it will be forwarded to the standard output of the tester.

For more information on using visualizers, please check the following recipe draft from TopCoder Cookbook. Note that this is not a troubleshooting thread, please use the match forum for questions instead.