CSCI 135 Programming Exam #0
Fundamentals of Computer Science I
Fall 2012

This part of the exam is like a mini-programming assignment. You will create a program, compile it, and debug it as necessary. This part of the exam is open book and open web. You may use code from the course web site or from your past assignments. When you are done, submit all your Java source files to the Moodle exam #0 dropbox. Please ***double check you have submitted all the required files***.

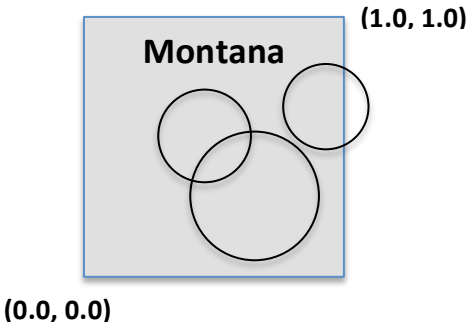
You will have 100 minutes. No communication with any non-staff members is allowed. This includes all forms of real-world and electronic communication.

Grading. Your program will be graded on correctness and to a lesser degree on clarity (including comments) and efficiency. You will lose a substantial number of points if your code does not compile or crashes on typical inputs.

Overview. You will be developing two programs `SnowSimulator.java` and `SnowStats.java`. The first generates random data about snow storms that have occurred this season in Montana. The second program takes input of the form generated by `SnowSimulator.java` and computes various statistics about the storms. It can also return the total season snowfall amounts at a specific location in the state. Stub versions of these two programs as well as test files and StdIn.java can be downloaded from here:

http://katie.mtech.edu/classes/csci135/snow.zip

SnowSimulator. Your first task is to develop a program that generates random snowstorm data. For the purposes of this problem, we assume the state of Montana is a unit square. A location in the state is given by two floating-point numbers x and y in `[0.0, 1.0)`, that is x and y are inclusive of `0.0` but exclusive of `1.0`. We assume storms are circles with a certain radius and always have their center somewhere inside the state (though as shown below, some storms may extend outside the state's border). Associated with each storm is a positive integer representing the depth of snow (in millimeters) delivered by that storm.



Your `SnowSimulator` program should take three command-line arguments:

1. ***Number of storms***, a positive integer specifying how many storms to generate.
2. ***Maximum storm radius***, a floating-point value (denoted here `maxStormRadius`) specifying the maximum radius a storm can have. Each generated storm has a random radius in `[0.0, maxStormRadius)`.
3. ***Maximum storm amount***, a positive integer (denoted here `maxStormAmount`) specifying the maximum snowfall amount. All snowfall amounts are reported in integer units. Each generated storm has a random snowfall amount in `[1, maxStormAmount]`. That is the snowfall is always at least 1 unit and can have up to an *including* `maxStormAmount` units of snowfall.

Your program outputs a line for each randomly generated storm. The format of each line is the x-location, y-location, radius, and amount of each storm. The 1st three numbers should be output as floating-point values, the last number as an integer. The x- and y-locations are randomly chosen in [0.0, 1.0).

Here is an example run:

```
% java SnowSimulator 5 0.5 10
0.7895825296514282 0.5451266663528596 0.34759357320640194 1
0.43297446545390594 0.9707094345265366 0.21765004910127955 10
0.23765995106571192 0.24625763933290756 0.18160927622540196 1
0.6483627385213898 0.39155287691944074 0.3974716103418983 4
0.7644457919280613 0.3972971968186496 0.03942274565587556 8
```

SnowStats – part 1. Create a program `SnowStats.java` that read in from standard input data of the form produced by `SnowSimulator`. Your program should calculate the following statistics about the storm data:

1. **Number of storms**, the integer count of the number of storms in the data file
2. **Minimum amount delivered by any storm**, the integer minimum snow delivered by any storm
3. **Maximum amount delivered by any storm**, the integer maximum snow delivered by any storm
4. **Average amount delivered by any storm**, that is the sum of all storm amounts divided by the number of storms.
5. **Total snow volume of all storms**, the total snow volume delivered by all storms. A particular storm delivers a volume equal to the area of the circle times the amount delivered by the storm. So the volume v of a storm with radius r that had a snow amount of a is given by: $v = \pi * r^2 * a$. Note in Java, the `Math` class has a constant for pi, `Math.PI`. You are to assume the total volume includes snow that has fallen outside Montana (due to storms on near the border).

Here is our output of SnowStats using the previously shown data (stored in `snow5.txt`) as well as several other input files:

```
% java SnowStats < snow5.txt
Storms: 5
Min storm amount: 1
Max storm amount: 10
Avg storm amount: 4.8
Total volume: 3.9957496213627857

% java SnowStats < snow10.txt
Storms: 10
Min storm amount: 188
Max storm amount: 961
Avg storm amount: 524.7
Total volume: 22.52359235955566

% java SnowStats < snow100.txt
Storms: 100
Min storm amount: 4
Max storm amount: 997
Avg storm amount: 487.66
Total volume: 828.0871119108149
```

SnowStats – part 2. Make your program support optionally calculating the total depth of snow that fell at a specific resort location in the state. The resort location is specified on the command-line by passing two arguments: the x-location and the y-location of the resort. The resort's coordinates are given by two floating-point values in [0.0, 0.0). If no command-line arguments are given, the program should continue to work as previously described in part 1.

If the resort location is given by (r_x, r_y) and a storm of radius r is located at (s_x, s_y), then the storm is considered to have dropped snow at the resort if the distance d between the resort and the storm is less than or equal to r (that is the resort's location is inside or on the storm's circle). Recall that you can calculate the Euclidean distance between two points using:

$$d = \sqrt{(r_x - s_x)^2 + (r_y - s_y)^2}$$

Your program should print out the total amount of snow that fell at the specified resort location (after reporting the statistics from part 1):

```
% java SnowStats 0.5 0.5 < snow5.txt
Storms: 5
Min storm amount: 1
Max storm amount: 10
Avg storm amount: 4.8
Total volume: 3.9957496213627857
Resort amount: 5

% java SnowStats 0.1 0.1 < snow5.txt
Storms: 5
Min storm amount: 1
Max storm amount: 10
Avg storm amount: 4.8
Total volume: 3.9957496213627857
Resort amount: 0

% java SnowStats 0.9 0.9 < snow100.txt
Storms: 100
Min storm amount: 4
Max storm amount: 997
Avg storm amount: 487.66
Total volume: 828.0871119108149
Resort amount: 915

% java SnowStats < snow100.txt
Storms: 100
Min storm amount: 4
Max storm amount: 997
Avg storm amount: 487.66
Total volume: 828.0871119108149
```