COT 4400: Analysis of Algorithm
Problem Modeling: Individual Project (100 pts)
Due: April 8, 2014

This project requires you to model the problem below as graph and then use a known graph algorithm to solve the problem.

You are not allowed to use the internet or consult any unapproved references[1]. This is an individual project. This policy will be strictly enforced.

1. *Problem Description*:

   This problem is based on the "Apollo and Diana" maze problem (from "MAD MAZES: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People" by Robert Abbott, Bob Adams, Inc. Publishers, 1990). The text of the problem is quoted below. A diagram of the maze is provided separately.

   > Apollo, god of the sun, stole a set of arrows from his sister Diana, goddess of the moon. When Diana caught him shooting moonbeams instead of sunbeams, she was furious! She cast a mighty curse, and all his arrows fell around him in a terrible maze. That maze is shown here. Apollo's sunbeams are the red arrows; Diana's moonbeams are the blue arrows.
   >
   > Diana spoke to Apollo: "You will remain here in this field of arrows until you solve this puzzle. You must find a route from the red arrow at the upper left to the bull's-eye at the lower right. The first red arrow points to two blue arrows. Choose either of those blue arrows and move to it. You are now on a blue arrow that points to one or more red arrows. Choose one of those red arrows. and move to it. Continue in this fashion, alternating between red and blue arrows. If you are truly wise you will arrive at an arrow that points to the bull's eye. You may then proceed to the bull's eye and escape from the maze. It's far more likely, though, that you will wander into a loop and find yourself going around endlessly. If that happens, you can admit that you are lost, go back to the red arrow at the upper left, and start the puzzle over again."
   >
   > So, how can Apollo find his way to the bull's eye (without getting stuck in too many loops)?

- The Input: (`input.txt`)
  The input file contains multiple instances. The input file begins with a single positive integer on a line by itself indicating the number of instances following, each of them as described below. This line is followed by a blank line, and there is also a blank line between consecutive instances.
  Each instance begins with a single positive integer on a line by itself indicating the dimension, n, of the maze. The following lines contain the directional and color information for each square of the maze. There are n lines which each have n values. Each value is represented as a direction followed by a dash (i.e., -) followed by the color. The direction is represented using `N, S, E, W, NE, NW, SE, SW` to represent "north", "south", "east", "west", "northeast", "northwest", "southeast", and "southwest", respectively. The color is represented using `R, B` to represent "red" and "blue", respectively. 0 represents the goals square (bull-eye).
  For the maze example provided, the input is:

```
  8
 E-R   SE-R   S-B   SW-B   S-R   SW-R   S-R   S-R
 E-B    S-R  SE-B   E-R   SE-B   S-B   W-B  SW-R
 N-R    W-B  SW-B  SE-R   NE-R  SW-B   W-B   W-R
SE-R   SE-R  SW-B  SE-R   S-R   NW-B   E-R   NW-B
NE-B    W-R   S-R   S-B   E-B   NE-B  NW-B  NW-R
 S-R   SE-B  SE-R  SE-R   NW-R  NE-R   E-B   W-R
NE-R    W-B  SE-B   E-R   E-R   E-B   NW-B  SW-R
NE-B    E-R   N-B  NE-R   NE-B   N-B  NW-B    0
```

  The maximum dimension for a maze is $n = 20$[2].

[1] You may only consult references listed on the approved references handout.

[2] If your algorithm/code is too slow to handle inputs of the prescribed size you will lose points for both your algorithm design grade AND your results grade

- The Output: (<`studentLastName`>`.txt`)
  The output file should contain one line for each instance. There should be a blank line between consecutive outputs.

  The output for each instance should consist of a path from the top left square to the bottom right square (bull-eye). There should be a single line of output which indicates the path Apollo should take. Each move should be represented by a direction and an integer representing the number of squares moved. The direction should be represented using `N, S, E, W, NE, NW, SE, SW` to represent "north", "south", "east", "west", "northeast", "northwest", "southeast", and "southwest", respectively. There should be a dash between the integer representing the number of squares moved and the direction. Each move should be separated by a space.

  Example: Suppose your first move takes you 5 places east to the square containing a blue arrow pointing southwest. Your second move takes to 4 places southwest to a square containing a red arrow pointing south. This sequence of two moves should be displayed on your output as follows:

  ```
  E-5 SW-4
  ```

A sample input/output file will be posted on blackboard.

You are welcome to try figuring out the solution to the puzzle on your own, but that won't get you any points. Your assignment is to model the maze as a graph and to solve the problem using an appropriate graph algorithm that we've encountered in class.

2. *Extra Credit Problem Description*:

   This problem is based on the "Apollo's Revenge" maze problem (from "MAD MAZES: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People" by Robert Abbott, Bob Adams, Inc. Publishers, 1990). The text of the problem is quoted below. A diagram of the maze is provided separately.

   > After several days, Apollo escaped from the arrow maze that his sister had placed him in, and after his long entrapment he was determined to have his revenge. He created his own maze out of arrows, and then hid nearby. When his sister came to check on her captive, he leapt out of hiding, overpowered her, and threw her into the maze of arrows.
   >
   > He cried out to her: "You will remain here until you solve this puzzle as I had to. As before, you must start on the red arrow at the upper left, travel from red arrow, to blue arrow, to red arrow, and so forth, until you come to an arrow that lets you move towards the bull's eye."
   >
   > "But that's not all," gloated Apollo, "Because I am so very clever, I added a new rule that will utterly confuse you. When you begin the maze, you move in the directions that the arrows point; that is, you move where the *head* of an arrow points. When you land on an arrow inside of a circle, you start moving in the *opposite* direction. Beginning at the arrow in the circle, move in the direction pointed to by the *tail* of the arrow. On subsequent arrows, continue moving where the tail points until you again stop on an arrow in a circle. At that point, switch back to moving where the *heads* point. Thus, each time you stop on an arrow in a circle, you change whether you follow heads or tails. Just passing *over* a circle has no effect on your movement."
   >
   > Is Apollo's new rule really that confusing? If not, can you find your way through this maze?

- The Input: (`input.txt`)
  The input file contains multiple instances. The input file begins with a single positive integer on a line by itself indicating the number of instances following, each of them as described below. This line is followed by a blank line, and there is also a blank line between consecutive instances.

  Each instance begins with a single positive integer on a line by itself indicating the dimension, n, of the maze. The following lines contain the directional and color information for each square of the maze. There are n lines which each have n values. Each value is represented as a direction followed by a dash (i.e., -) followed by the color. If the arrow is inside a circle the direction and color will be followed by a dash (i.e., -) and the character C. The direction is represented using `N, S, E, W, NE, NW, SE, SW` to represent "north", "south", "east", "west", "northeast", "northwest", "southeast", and "southwest", respectively. The color is represented using `R, B` to represent "red" and "blue", respectively. 0 represents the goals square (bull-eye).

  For the maze example provided, the input is:

```
 7
E-R    W-B    NW-B   NW-R   S-R   SW-B   S-B-C
SE-R   S-B    SE-R    S-B   N-R   NW-R   NE-R
E-B   N-R-C  S-R-C   E-B    N-B    S-B   NE-B
N-R   NE-R   E-B-C   SW-R   N-R    E-B   SW-B
W-B    S-R   W-B-C   SW-R   S-B   SE-B   NE-R
SW-R   S-B    NW-B   E-R-C  SE-R   S-R    S-B
SW-B   N-R     E-B    SW-R   SE-B  NW-B     0
```

  The maximum dimension for a maze is $n = 20^3$.

- The Output: (<`studentLastName`>`.txt`)
  The output file should contain one line for each instance. There should be a blank line between consecutive outputs.

  The output for each instance should consist of a path from the top left square to the bottom right square (bull-eye). There should be a single line of output which indicates the path Apollo should take. Each move should be represented by a direction and an integer representing the number of squares moved. The direction should be represented using `N, S, E, W, NE, NW, SE, SW` to represent "north", "south", "east", "west", "northeast", "northwest", "southeast", and "southwest", respectively. There should be a dash between the integer representing the number of squares moved and the direction. Each move should be separated by a space.

  Example: Suppose your first move takes you 5 places east to the square containing a blue arrow pointing southwest. Your second move takes to 4 places southwest to a square containing a red arrow pointing south. This sequence of two moves should be displayed on your output as follows:

```
E-5 SW-4
```

You are welcome to try figuring out the solution to the puzzle on your own, but that won't get you any points. Your assignment is to model the maze as a graph and to solve the problem using an appropriate graph algorithm that we've encountered in class.

[3] If your algorithm/code is too slow to handle inputs of the prescribed size you will lose points for both your algorithm design grade AND your results grade

3. *Deliverables*:

   Please submit a **hard copy** of all of the items requested below. Please don't send this by email as I'll have to print it out anyway. Please also submit *code*[4] and *typed project report*[5] to Canvas.

   (a) **Problem Modeling [50 pts]**:

   - **[15 pts]**: Explain how you modeled the problem as a graph.
     For full credit your explanation must be descriptive enough that any competent programmer will be able to create a graph of any valid input based on your description and understand the reasoning behind how the graph is created.
   - **[10 pts]**: Draw the resulting graph.
     For full credit you should draw the resulting graph for the instance of the maze provided by the problem statement. This may be hand-drawn
   - **[10 pts]**: Identify the known graph algorithm needed to solve the problem.
   - **[15 pts]**: Prove that this algorithm will actually solve the problem.
     For full credit you must provide a formal proof that this algorithm will produce the correct solution for any valid input to the problem (i.e., you will need to prove that how you modeled the problem as a graph accurately represents the problem and that the algorithm you identified will produce the correct solution on this graph.)

   (b) **Code [20 pts]**:

   Submit a print-out of your code. If you are unable to get your code to compile/run, please state this explicitly. Code Requirements[6]:

   - Read the input from a file in the described format.
   - Output the solution in the described format.
   - README file describing how to compile and run your code. If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script.
     A simple compilation command should only include the compiler to be used and the name of a single file. A simple run command should only include the name of the executable. If your command requires any flags or directives then you must provide a Makefile and/or shell script. If you require input/output to be redirected from the console then you must provide a shell script. For example `g++ main.cpp` *is a simple command.*
   - *Optional: You may also submit an executable for the C4 Linux Lab machines*
   - You must use an algorithm from an approved graph algorithms library.

   For full credit your code must follow all of the listed requirements.

   (c) **Results [30 pts]**:

   Submit your code (in addition to your code you may submit an executable) to Blackboard to be run on a set of inputs devised by the grader. Your grade will be based on whether your code produces the correct output for each of the inputs devised by the grader.

   (d) **Penalty [-10pts]**: If you choose to implement the known graph algorithm yourself then a 10 point penalty will be assessed.

   (e) **Extra Credit [10pts]**: Extend your *implementation and report* to solve the "Apollo's Revenge" maze problem (from "MAD MAZES: Intriguing Mind Twisters for Puzzle Buffs, Game Nuts and Other Smart People" by Robert Abbott, Bob Adams, Inc. Publishers, 1990). Note that you are only eligible to earn the extra credit if you are using a known graph algorithm implemented by an approved graph algorithms library (i.e., you have not taken the 10 point penalty above).

[4] Your code must compile and run on the C4 Linux Lab machines
[5] Project report and code will be checked for plagiarism
[6] Failure to follow these requirements may result in a 0 for the code portion of your grade AND a 0 in the results portion of your grade