COP 4530: Data Structures
Project 6

You are not allowed to use the Internet. You may only consult approved references*.
This is a group project.
This policy is strictly enforced.

You must submit a **hard copy** of all of the items requested below. You must also submit your *code*† to Canvas.

If you would like to work in a group, then the following rules apply:

1. Groups may have at most 3 students
2. Each student must implement a different algorithm to find a path through the maze (all students in the group should use the same graph data structure and code to create the graph)
3. You must "register" your group on Canvas (this is done in the "People" section of Canvas)

For full credit, the code that is submitted must:

- Use the specified signature, if applicable.
- Be implemented in a file using the specified file name, if applicable.
- Be correct (i.e., it must always return the correct result).
- Be efficient (i.e., it must use the minimum amount of time and the minimum amount of space necessary to be a correct implementation).
- Be readable and easy to understand. You should include comments to explain when needed, but you should not include excessive comments that makes the code difficult to read.
  - Every class definition should have an accompanying comment that describes what is for and how it should be used.
  - Every function should have declarative comments which describe the purpose, preconditions, and postconditions for the function.
  - In your implementation, you should have comments in tricky, non-obvious, interesting, or important parts of your code.
  - Pay attention to punctuation, spelling, and grammar.
- Follows ALL coding guidelines from section 1.3 of the textbook. Additional coding guidelines:
  - No magic numbers. Use constants in place of hard-coded numbers. Names of constants must be descriptive.
  - No line of the text of your source code file may have more than 80 characters (including whitespace).
  - All header files should have #define guards to prevent multiple file inclusion. The form of the symbol name should be `<FILENAME>_H_`
  - Do not copy and paste code. If you need to reuse a section of code, then write a function that performs that code.
  - Define functions inline only when they are simple and small, say, 5 lines or less
  - Function names, variable names, and filenames must be descriptive. Avoid abbreviation.
  - Use only spaces (no tabs), and indent 3 spaces at a time.
- Compile and run on the C4 Linux Lab machines (g++ compiler, version 4.8.2). *The shell script and makefile that I will use to compile and run your code will be posted on Canvas. Please note that I may use my own* `main.cpp` *file to test the code you submit.*
- Have no memory leaks.

*The list of approved references is posted on Canvas. You must cite all references used.
†Your code must compile and run on the C4 Linux Lab machines

Project Description

This problem is based on the "Jumping Jim's Encore" 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.

> Our story so far: In Maze 7, Dastardly Dan had tried to sabotage Jumping Jim's act by restringing all his trampolines. But Dan's actions had the opposite effect. The audience was so delighted by Jim's leaping about trying to reach the goal, that his act became the most popular of the circus.
>
> The circus owner decided to commission Dastardly Dan to create another, even harder maze for Jim to solve. Dan added more trampolines, restrung them, and painted large numbers on each to indicate how far Jim will move when he bounces off each trampoline. The painted numbers would allow the audience to study the maze and try to find a solution before Jim did.
>
> Dan also added a new rule that Jim had to follow. Certain of the numbers were painted in red and enclosed in circles. When Jim begins his act, he can move only vertically or horizontally through the maze of trampolines; he cannot move diagonally. However, if he lands on a red number in a circle, he must then start moving only diagonally; now he can't move vertically or horizontally. Jim must continue moving diagonally until he again lands on a red number. He then switches back to moving only vertically or horizontally. And he switches each time he lands on a red number.
>
> Here's an example to show how that works. Jim begins on the 4 at the northwest corner of the maze. From there he might move south four squares to the red 3. Now he must start moving diagonally. He might go three squares northeast to a 4. On the next move he would continue moving diagonally. He could move four squares southeast to a red 1. That red number would cause him to switch back to moving only horizontally or vertically.
>
> Can you find a route that would let Jim land on the trampoline marked GOAL?

- 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 values for each square of the maze. There are n lines which each have n integer values. 0 represents the goals square. If it is a red square then it is a negative integer. For the maze example provided, the input is:

```
8
 4   2  -2   4   4  -3   4  -3
 3   5   3   4   2   3   5  -2
 4   3   2  -5   2   2   5   2
 7   1   4   4   4   2   2   3
-3   2   2   4   2   5   2   5
 2  -3   2   4   4   2   5  -1
 6   2   2  -3   2   5   6   3
 1  -2   5   4   4   2  -1   0
```

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

  Each output should consist of a path from the top left square to the bottom right square. There should be a single line of output which indicates the path Jim should take. Each move should be represented using `N, S, E, W, NE, NW, SE, SW` to represent "north", "south", "east", "west", "northeast", "northwest", "southeast", and "southwest", respectively. Each move should be separated by a space.

  Example: Suppose your first move takes you 4 places south to the square marked 3R and your second move takes you northeast to the square marked 4. This sequence of two moves should be displayed on your output as follows:

```
S NE
```

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.

Project Tasks

1. [30 points] Implement a data structure to hold a directed graph.
   For full credit you:
   (a) must use an adjacency list as its underlying data structure.
   (b) should use the C++ STL containers to hold your adjacency list.
   (c) are not permitted to modify the data structure from the description provided in lecture (i.e., your graph must have one set of vertices and one set of edges - you are not permitted to distinguish between different types of edges or different types of vertices.)

2. [35 points] Implement an algorithm to model the maze as a graph (using your graph data structure from task 1 above)
   If the maze is an $n \times n$ maze, then you should allocate $2n^2$ vertices. The vertices 0 through $n^2 - 1$ are used to horizontal/vertical movement. The vertices n^2 through $2n^2 - 1$ are used for diagonal movement. When you land on a red square, you should have directed edges to the appropriate position in the opposite "movement directions" of the maze. To determine where you are in the maze (i.e., row/column) you should use division and modulus operations.

3. [20 points] Each group member[‡] must implement a unique algorithm (selected from below) to find a path through the graph used to model this maze.
   (a) Breadth-First Search
   (b) Depth-First Search
   (c) Dijkstra's Algorithm

4. [15 points] Write one main function that reads an input file in the specified format and outputs a solution for each maze in the specified format[§].

5. Submit group effort percentages and describe who did what. The effort percentages must be agreed upon by the group. See syllabus to determine how we will use this to compute your individual grade.
   Note that we reserve the right to change your group effort percentages. Group effort percentages will be changed if we think they do not accurately represent the contribution of each student.

[‡] You must clearly indicate which group member implemented each algorithm.
[§] Create one output file for each algorithm implemented in Task 3 above.