Programming Assignment 4
CSCE 221

1. The description of an assignment problem.
   (a) The maze program is a program that gathers input that describes a maze and stores the information in a vector. This program then takes the created vector and creates an adjacency list which is used to find the shortest path from the entrence to the exit in a maze, using the Depth First Search Algorithm.
2. The description of data structures and algorithms used to solve the problem.
   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)
      i. The data structures that this program uses is based on a class to create an adjacency matrix and use the DFS algorithm, and vectors which strores the maze input. The depth-first search algorithm uses a stack to keep track of all of the rooms visited and the path to solve the maze.
   (b) Write about the ADTs implementation in C++.
      i. The vectors are implemented in the defining of each member function and creating containers to hold data also, the stack is used to traverse a two-dimensional vector to find a path from the start to the end of a given maze.
   (c) Describe algorithms used to solve the problem.
      i. DFS: This algorithm is used to find the shortest path from the entrance to the exit of a given maze, by exploring all verticies until a path is determined.
   (d) Analyze the algorithms according to assignment requirements.
      i. The worst case scenarios for the DFS algorithms is O(n+m) and this happens when each vertex must be visited to find the solution.

3. A C++ organization and implementation of the problem solution

   (a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

      i. class Graph:

         A. constructors: allocates and initializes room and pathLength integers
         B. readMaze: reads maze information and sets adjMatrix
         C. DFS: finds shortest path from entrance to exit
         D. printMaze: prints maze solution using characters
         E. printMatrix: prints adjacency matrix
         F. printPathLenght: print length of the solution path
         G. findPathSolution: uses DFS to return the solution path of the maze

   (b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

      i. Part 1: Maze
      maze.h
      maze.cpp
      maze.txt
      maze2.txt
      Makefile

   (c) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

      i. Polymorphism is present in mulitply member functions where these size of the vector is needed.

4. A user guide description how to navigate your program with the instructions how to:

   (a) compile the program: specify the directory and file names, etc.

      i. Part 1: My_vec

         A. cd maze
         maze.h
         maze.cpp
         maze.txt
         maze.o
         Makefile
         B. make all

   (b) run the program: specify the name of an executable file.

      i. ./maze

5. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

      i. keyboard input for maze input file.

   (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

      i. text file needed

   (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

      i. Error when menu choice has incorrect output.

```
Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit

Your choice: 5


invalid input ... please try again


Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit
```

6. Provide types of exceptions and their purpose in your program.

   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

      i. Exception thrown when menu choice has incorrect output.

   (b) runtime exception (such as division by 0, etc.)

      i. Exception thrown when menu choice has incorrect output, or incorrect text file is inputed

7. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

   (a) Maze

```
Owners-MacBook-Air:finalProject owner$ make all
c++ -std=c++11 Maze.o -o maze
Owners-MacBook-Air:finalProject owner$ ./maze
Enter name of maze file: maze.txt
mazeVec:
0: 0 1 0 0
1: 0 1 1 1
2: 0 1 1 1
3: 0 0 0 1
4: 0 1 0 0
5: 1 0 1 1
6: 1 0 1 0
7: 0 0 1 0
8: 0 1 1 0
9: 1 0 0 1
10: 1 1 0 0
11: 1 0 0 1
12: 1 1 0 0
13: 0 1 0 1
14: 0 1 0 1
15: 0 0 0 1

DFS: 0 1 5 9 8 12 13 14 15

Maze Solution:
o o x x
x o x x
o o x x
o o o o

Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit

Your choice: 0


Matrix
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit

Your choice: 1


Path Solution: 0 1 5 9 8 12 13 14 15
```

```
Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit

Your choice: 2


Path Length: 9

Enter '0' to see adjacency matrix
Enter '1' to see rooms in the solution path
Enter '2' to see the length of the solution path
Enter '3' to exit

Your choice: 3


EXITING
Owners-MacBook-Air:finalProject owner$
```