**Jeff Morton**

**Data Structures and Algorithms 2**

**Project 1**

**Functional Decomposition**

**User-defined data structures used as parameters in the functions:**

typedef struct Graph

{

int graphSize, width;

int \*\*adjacencyMatrix;

}Graph, \*GraphP;

**Files and Functions in the Program:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Student Name: Jeff Morton

\* File Name: Graph.c

\* Assignment Number: 3

\* Date Due: Mar 13, 2016

\*

\* Created on: Mar 7, 2016

\* Author: Jeff Morton (jhm14@students.uwf.edu)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Graph.c (Use for Graph.h as well, same functions)**

/\*\*

\* initializes all the elements of a an adjacency matrix for a grid graph

\* @param {\*\*matrix} matrix - the adjacency matrix

\* @param {int} graphSize - the size of the graph

\* @param {int} width - the width of the graph

\*/

**void initializeAdjacencyMatrix( int \*\*matrix, int graphSize, int width);**

/\*\*

\* Creates, allocates, and initializes a new graph

\* @param {int} graphSize - the size of the graph

\* @param {int} width - the width of the graph

\* @return GraphP the newly created graph

\*/

**GraphP newGraph( int graphSize, int width);**

/\*\*

\* initializes all the elements of a an adjacency matrix for a grid graph

\* @param {\*\*matrix} matrix - the adjacency matrix

\* @param {int} graphSize - the size of the graph

\* @param {int} width - the width of the graph

\*/

**void initializeAdjacencyMatrix( int \*\*matrix, int graphSize, int width);**

/\*\*

\* prints out the graph passed in csv format.

\* @param {GraphP} graph - the graph to print

\*/

**void printGraph( GraphP graph);**

/\*\*

\* Creates a MST of the graph passed using Prim's Algorithm

\* @param {GraphP} graph - the graph to generate a MST for

\*/

**void primsAlgorithm( GraphP graph);**

/\*\*

\* Checks to see if there is a connection between the 2 vertices passed in the graph passed

\* @param {GraphP} graph - the graph to search

\* @param {int} vert1 - the first vertex

\* @param {int} vert2 - the second vertex

\*/

**int isLink( GraphP graph, int vert1, int vert2);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Student Name: Jeff Morton

\* File Name: Maze.c

\* Assignment Number: 3

\* Date Due: Mar 13, 2016

\*

\* Created on: Mar 9, 2016

\* Author: Jeff Morton (jhm14@students.uwf.edu)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Maze.c (Use for Maze.h as well, same functions)**

/\*\*

\* Prints out the maze to the file pointer passed

\* @param {GraphP} graph - the graph to print the maze from

\* @param {FILE \*} outFile - the file to print to

\*/

**void printMaze( GraphP graph, FILE \*outFile);**

/\*\*

\* Prints out the maze to Maze.ppm in PPM format

\* @param {GraphP} graph - the graph to print the maze from

\*/

**void printPPM( GraphP graph);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Student Name: Jeff Morton

\* File Name: Main.c

\* Assignment Number: 3

\* Date Due: Mar 13, 2016

\*

\* Created on: Mar 7, 2016

\* Author: Jeff Morton (jhm14@students.uwf.edu)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Main.c**

/\*\* Returns the square root of the number passed if it's a perfect square, or 0 if not.

\* @param {int} num - the number to find the square root of

\* @return int either the square root or 0 if not a perfect square

\*/

**int getSqrt(int num);**

/\*\*

\* the main function

\*/

**int main(int argc, char \*argv[]);**