**Programmer Manual**

**Network ADT Program**

1. **Problem Description**

This program presents the viewer with a menu displaying 14 different options. Its purpose is to allow for a graph construction or destruction of any type, and size. Error messages are given if the inputs received are not any of the displayed options. The program is written to input a graph from a user-file; this program will accept formatted .txt files only. Any other file will produce incorrect results.

1. **Data Types and Classes**

The data types used in this program are of two categories: pre-defined and programmer defined.

* 1. **int (predefined)**

Variables:

choice – to hold the integer value of which the user entered

weight – to hold the weight of user input data

* 1. **string (predefined)**

Variables:

v1 – the name of a vertex

v2 – the name of a vertex

* 1. **Graph (programmer defined)**

This class has:

Data members: G

Member functions: isVertex, isUniEdge, isBirDirEdge, AddVertex, DeleteVertex, AddUniEdge, DeleteUniEdge, AddBiDirEdge, DeleteBiDirEdge, SimplePrintGraph, FordShortestPaths, BFTraversal, DFTraversal, GetGraph, mst, DepthFirstTraversal, DFT

Variables: graph – object used to access functions of the Graph class

1. **High Level Program Solution**

**Main Program**

Prompt user for option selection for directive.

IF choice 1

Calls GetGraph function to get the graph’s file path from the user.

Asks user to select another input.

IF choice 2

Asks user to input a vertex.

Calls AddVertex function to add a new vertex to the graph.

Asks user to select another input.

IF choice 3

Asks user to input a vertex.

Calls DeleteVertex function to delete an existing vertex.

Asks user to select another input.

IF choice 4

Asks user to input a source vertex, destination vertex, and its edge’s weight.

Calls AddUniEdge function to add a unidirectional edge to the graph.

Asks user to select another input.

IF choice 5

Asks user to input a source vertex, destination vertex.

Calls DeleteUniEdge function to remove a unidirectional edge from the graph.

Asks user to select another input.

IF choice 6

Asks user to input a source vertex, destination vertex, and its edge’s weight.

Calls AddBiDirEdge function to add a bidirectional edge to the graph.

Asks user to select another input.

IF choice 7

Asks user to input a source vertex, destination vertex.

Calls DeleteBiDirEdge function to remove a bidirectional edge from the graph.

Asks user to select another input.

IF choice 8

Calls SimplePrintGraph function to print the entire graph.

Asks user to select another input.

IF choice 9

Asks user to input a source vertex, destination vertex.

Calls ShortestDistance function to print the shortest path between the two vertices in the graph.

Asks user to select another input.

IF choice 10

Calls mst function to create and print a minimum spanning tree for the graph.

Ask user to select another input.

IF choice 11

Asks user to input a source vertex.

Calls BFTraversal function to print a breadth first traversal of the graph.

Asks user to select another input.

IF choice 12

Asks user to input a source vertex.

Calls DFTraversal function to print a depth first traversal of the graph.

Asks user to select another input.

IF choice 13

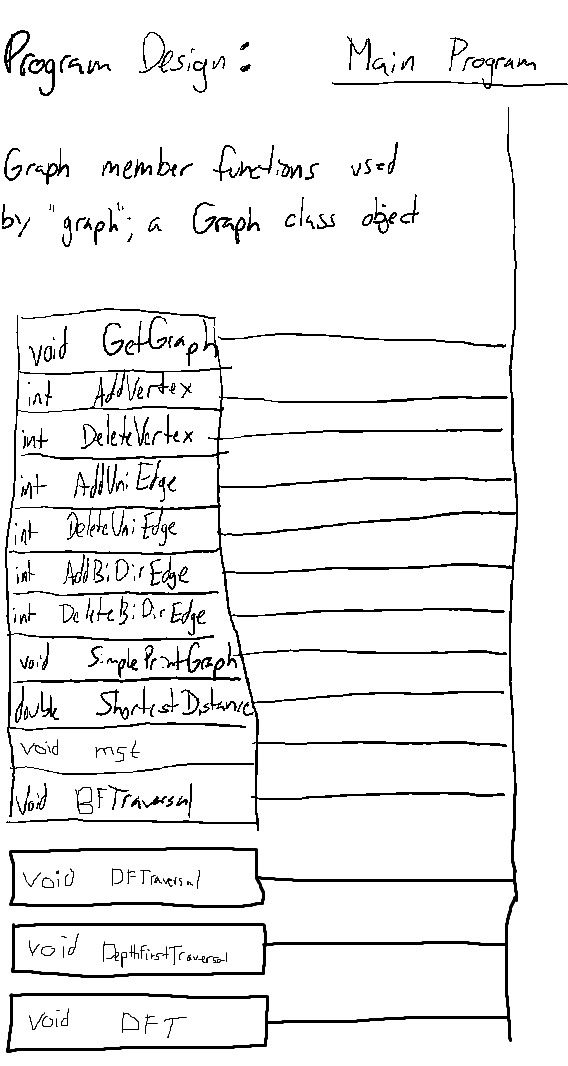
Asks user to input a source vertex.

Calls DFT function to print a recursive depth first traversal of the graph.

Asks the user to select another input.

IF choice 14

Ends the program.



1. **Limitations and Suggestions**

This program is reliant upon a properly formatted user named input file. This is a potential problem if one wishes to read in data from any txt file. The program is written to handle any type of vertex, but slight modifications to code will be necessary in order to achieve desired results.