**Programmer Manual**

**Graph ADT Program**

1. **Problem Description**

This program presents the viewer with a menu displaying 12 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 as follows: 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, ShortestDistance, BFTraversal, DFTraversal, GetGraph

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

ask user to select further instruction.

If choice 2

Asks user to enter vertex.

Calls AddVertex function to add a new vertex into the graph

ask user to select further instruction.

IF choice 3

Asks user to enter vertex.

Calls DeleteVertex function to delete an existing vertex

ask user to select further instruction.

IF choice 4

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

Calls AddUniEdge function to add a unidirectional edge to the graph

ask user to select further instruction.

IF choice 5

Asks user to enter source vertex, destination vertex.

Calls DeleteUniEdge function to remove a unidirectional edge from the graph

asks user to select further instruction.

IF choice 6

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

Calls AddBiDirEdge function to add a bidirectional edge to the graph

asks user to select further instruction.

IF choice 7

Asks user to enter source vertex, destination vertex.

Calls DeleteBiDirEdge function to remove a bidirectional edge from the graph

asks user to select further instruction.

IF choice 8

Calls SimplePrintGraph function to print the entire graph

asks user to select further instruction.

IF choice 9

Asks user to enter source vertex, destination vertex.

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

asks user to select further instruction.

IF choice 10

Asks user to enter source vertex.

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

asks user to select further instruction.

IF choice 11

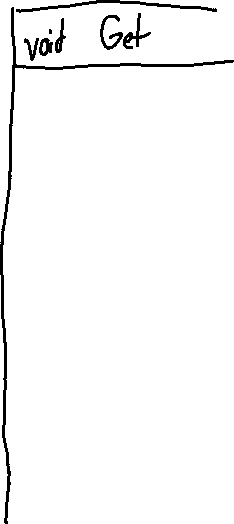
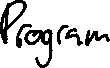
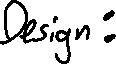
Asks user to enter source vertex.

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

asks user to select further instruction.

IF choice 12

ends the program.



1. **Limitations and Suggestions**

This program is partly reliant on 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.