1. Download the following files and create a workspace using them:

<https://markbowman.org/231/Lab24.zip>

The code file contains global arrays with node names and edge information. Use them to complete the lab, but don’t create any other global variables.

2. Compile and run the program. You should see this displayed:

Sample Run

A B C D E

A -1.00 2.00 1.70 -1.00 -1.00

B -1.00 -1.00 3.50 1.50 -1.00

C -1.00 -1.00 -1.00 2.40 -1.00

D -1.00 -1.00 -1.00 -1.00 4.10

E 9.99 -1.00 -1.00 -1.00 -1.00

Text

Description automatically generated

3. Write code that will prompt the user for a source name, then set source\_index to the matching string in the name[] array.

4. Write code that will prompt the user for a target name, then set target\_index to the matching string in the name[] array.

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

\* Lab24.cpp

\* Twymun K. Safford

\* Last Updated: 11/17/2021

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

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

#define GRAPH\_MAX 10

void display(int n);

string name[GRAPH\_MAX] = { "A","B","C","D","E" };

float edge[GRAPH\_MAX][GRAPH\_MAX] =

{ { -1.00, 2.00, 1.70,-1.00,-1.00 },

{ -1.00,-1.00, 3.50, 1.50,-1.00 },

{ -1.00,-1.00,-1.00, 2.40,-1.00 },

{ -1.00,-1.00,-1.00,-1.00, 4.10 },

{ 9.99,-1.00,-1.00,-1.00,-1.00 }

};

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

\* main()

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

void main()

{

int i, n;

int source\_index, target\_index;

string s, t; //source and target respectively

float distance, total;

// Initialize

n = 5;

// Display

display(n);

cout << endl;

//prompt user to enter source node

cout << "Enter source: ";

cin >> s;

// extract the first character from the input string index and set source\_index

source\_index = s[0] - 'A';

//prompt user for target node

cout << "Enter target: ";

cin >> t;

// extract the first character from the input string index and set target\_index

target\_index = t[0] - 'A';

//if-else statements to calculate and display the distance between nodes - creates warning if no valid edge

//if the submitted indices are valid (0 to 5):

if (target\_index >= 0 && target\_index < n && source\_index >= 0 && source\_index < n)

{

//no valid edge from the source to the destination

if (edge[source\_index][target\_index] < 0)

{

cout << "No edge from " << (char)('A' + source\_index) << " to " << (char)('A' + target\_index) << endl;

}

//if there is a valid distance from the source to the destination along a valid edge, display source to destination

else

{

//print to screen the distance

cout << (char)('A' + source\_index) << " to " << (char)('A' + target\_index) << " = " << edge[source\_index][target\_index] << endl;

}

}

//for wrong input - tell user

else

{

cout << "Wrong input. Try again" << endl;

}

return;

}

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

\* display()

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

void display(int n)

{

int i, j;

// Initialize

cout << fixed << setprecision(2);

// Target row

for (j = 0; j < n; j++)

cout << setw(7) << name[j];

cout << endl;

// Loop through Sources

for (i = 0; i < n; i++)

{

cout << setw(2) << name[i]; // Name

for (j = 0; j < n; j++)

cout << setw(7) << edge[i][j]; // Edge distance

cout << endl;

};

}

5. Using source\_index, target\_index, and the edge[][] array, display the distance between the nodes, or a warning if there is no valid edge.

Sample Runs

A B C D E

A -1.00 2.00 1.70 -1.00 -1.00

B -1.00 -1.00 3.50 1.50 -1.00

C -1.00 -1.00 -1.00 2.40 -1.00

D -1.00 -1.00 -1.00 -1.00 4.10

E 9.99 -1.00 -1.00 -1.00 -1.00

Enter source: A

Enter target: C

A to C = 1.70

A B C D E

A -1.00 2.00 1.70 -1.00 -1.00

B -1.00 -1.00 3.50 1.50 -1.00

C -1.00 -1.00 -1.00 2.40 -1.00

D -1.00 -1.00 -1.00 -1.00 4.10

E 9.99 -1.00 -1.00 -1.00 -1.00

Enter source: B

Enter target: E

No edge from B to E

6. Test your program with three valid and one invalid edge, and save the outputs.

**Valid Outputs:**

Text

Description automatically generated

Text

Description automatically generated

**Text

Description automatically generated**

**Invalid Output:**

**Text

Description automatically generated**

7. Create a loop that will allow the user to move from node to node along valid edges and sum up the total distance. Prompt the user for a starting node, then loop until they enter a target of "Quit". Do not use a break to exit the loop.

Sample Run

A B C D E

A -1.00 2.00 1.70 -1.00 -1.00

B -1.00 -1.00 3.50 1.50 -1.00

C -1.00 -1.00 -1.00 2.40 -1.00

D -1.00 -1.00 -1.00 -1.00 4.10

E 9.99 -1.00 -1.00 -1.00 -1.00

Enter source: A

Enter target: C

A to C = 1.70

Enter target: D

C to D = 2.40

Enter target: B

No edge from D to B

Enter target: E

D to E = 4.10

Enter target: Quit

Total distance = 8.20

8. Test your program two times, with valid and invalid edges, and save the outputs.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Lab24 – Main.cpp:

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

\* Lab24.cpp - Extra Credit

\* Twymun K. Safford

\* Last Updated: 11/17/2021

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

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

#define GRAPH\_MAX 10

void display(int n);

string name[GRAPH\_MAX] = { "A","B","C","D","E" };

float edge[GRAPH\_MAX][GRAPH\_MAX] =

{ { -1.00, 2.00, 1.70,-1.00,-1.00 },

{ -1.00,-1.00, 3.50, 1.50,-1.00 },

{ -1.00,-1.00,-1.00, 2.40,-1.00 },

{ -1.00,-1.00,-1.00,-1.00, 4.10 },

{ 9.99,-1.00,-1.00,-1.00,-1.00 }

};

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

\* main()

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

void main()

{

int i, n;

int source\_index, target\_index;

string s; //source

string t; //target

float distance, total;

//total distance for while loop

float total\_distance;

// Initialize

n = 5;

total\_distance = 0.00;

// Display

display(n);

cout << endl;

//prompt the user for the source name

cout << "\nEnter source : ";

cin >> s;

//extract the very first character from the input string index and set source\_index

source\_index = s[0] - 'A';

//check if the source index or s are not acceptable

//original placement of target prior to moving into while loop

//boolean variable - assess if the user wants to quit

bool doNotQuit = true;

//do this loop as long as doNotQuit is true

do

{

//prompt the user for the target name

cout << "Enter target : ";

cin >> t;

if (t == "Quit")

{

//set boolean variable to quit

doNotQuit = false;

//print out the total distance travelled

cout << endl << "Total distance = " << total\_distance << endl;

//quit the program

return;

}

//extract the very first character from the input string index and set as target\_index

target\_index = t[0] - 'A';

//if-else statements to calculate and display the distance between nodes - creates warning if no valid edge

//if the submitted indices are valid (0 to 5):

if ((t.length() == 1) && (target\_index >= 0) && (target\_index < n) && (source\_index >= 0) && (source\_index < n))

{

//no valid edge from the source to the destination

if (edge[source\_index][target\_index] < 0)

{

cout << "No edge from " << (char)('A' + source\_index) << " to " << (char)('A' + target\_index) << endl;

}

//if there is a valid distance from the source to the destination along a valid edge, display source to destination

else

{

//calculate the new total distance based on addition of all weights between connected nodes

total\_distance = total\_distance + edge[source\_index][target\_index];

cout << (char)('A' + source\_index) << " to " << (char)('A' + target\_index) << " = " << edge[source\_index][target\_index] << endl;

//now, the new source becomes the old target and the new target needs to be redefined

source\_index = target\_index;

}

}

//if the user has submitted wrong input

else

{

//print out that the user needs to submit valid input

cout << "Wrong input. Try again" << endl;

}

//return statement

} while (doNotQuit == true);

//print out the total distance

return;

}

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

\* display()

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

void display(int n)

{

int i, j;

// Initialize

cout << fixed << setprecision(2);

// Target row

for (j = 0; j < n; j++)

{

cout << setw(7) << name[j];

}

cout << endl;

// Loop through Sources

for (i = 0; i < n; i++)

{

cout << setw(2) << name[i]; // Name

for (j = 0; j < n; j++)

{

cout << setw(7) << edge[i][j];

}// Edge distance

cout << endl;

};

return;

}

Extra Credit (3 points)

Update your code to allow the user to add additional nodes and edges to the graph. Test the program with at least one new node and two new edges. Save output from two runs, using these additions to the graph structure.

Create a MS-Word document and paste a copy of your code, and the sample runs. Your code will be graded on its functionality, appearance, and comments.