Write a program that will compute the distance from one city to another, using the map below. Create a two-dimensional table to store the distances.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Lansing | Flint | Detroit | Jackson | Grand Rapids | Battle Creek | Kalamazoo | Clare |
| Lansing | 0 | 53 | 80 | 34 | 75 | 35 | 0 | 85 |
| Flint | 53 | 0 | 77 | 0 | 0 | 0 | 0 | 94 |
| Detroit | 80 | 77 | 0 | 68 | 0 | 0 | 0 | 0 |
| Jackson | 34 | 0 | 68 | 0 | 0 | 30 | 0 | 0 |
| Grand Rapids | 75 | 0 | 0 | 0 | 0 | 0 | 59 | 109 |
| Battle Creek | 35 | 0 | 0 | 30 | 0 | 0 | 47 | 0 |
| Kalamazoo | 0 | 0 | 0 | 0 | 59 | 47 | 0 | 0 |
| Clare | 85 | 94 | 0 | 0 | 109 | 0 | 0 | 0 |

Download the following code files to save some typing:

<https://markbowman.org/231/Lab27.zip>

**(Messy)representation:**

Use the following algorithm:

* Prompt the user for **start** and **stop**, and initialize **total**.
* Assign **pos** equal to **start**. This will represent the current row.
* While **pos** is not equal to **stop**, and **pos** is less than **CITY\_MAX**, do the following:
* Use the index **temp** to find the first element in the row that is not 0.
* If **temp** is less than **CITY\_MAX**:

Display and add the distance in the array to your total.

Mark the element in the row as invalid (set the array element to 0).

* Assign **pos** equal to **temp**.
* Display the **total** distance.

Display the list of cities and distances. If the algorithm ends up at the destination, display the total distance, if not, display an appropriate message.

Sample Run

City List

----------------

0) Lansing

1) Flint

2) Detroit

3) Jackson

4) Grand Rapids

5) Battle Creek

6) Kalamazoo

7) Clare

Enter start city: 7

Enter stop city: 5

Clare to Lansing = 85

Lansing to Flint = 53

Flint to Lansing = 53

Lansing to Detroit = 80

Detroit to Lansing = 80

Lansing to Jackson = 34

Jackson to Lansing = 34

Lansing to Grand Rapids = 75

Grand Rapids to Lansing = 75

Lansing to Battle Creek = 35

Total = 604

**Screenshot:**

Text

Description automatically generated/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab27.cpp

\* Written by Twymun K. Safford

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

#include <iostream>

#include <string>

using namespace std;

#include "Lab27.h2"

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

\* main()

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

void main()

{

int i; //index counter

//start - starting city from user selection

//stop - end city from user selection

//temp - second index counter to seatch through all elements of 2d array for cities

int pos, temp, start, stop;

//total distance travelled

int total = 0;

//print statement for the city list with index

std::cout << "City List\n--------------------" << endl;

for (i = 0; i < CITY\_MAX; i++)

{

std::cout << i << ") " << city[i] << endl;

}

//ask user to input start city

std::cout << "\nEnter the index of the starting city: ";

cin >> start;

//ask user to enter output city

std::cout << "Enter the index of the stop city: ";

cin >> stop;

//set i equal to start and temp to 0

pos = start;

// starting city is always visited

int vis[CITY\_MAX] = { 0 };

vis[start] = 1;

//set temp to 0

temp = 0;

// make an array which will be used as flag to cities

// initially all cities are unvisited so marked 0

// visited city will be marked as 1

//iterate until start does not equal the end

while (pos != stop)// && pos < CITY\_MAX)

{

//for (i = 0; i < CITY\_MAX; i++)

//{

// map[i][temp] != 0 as well as the destination city is not visited

if ((map[pos][temp] != 0) && (vis[temp] == 0))

{

std::cout << city[pos] << " to " << city[temp] << " = " << map[pos][temp] << endl;

//add the weight for each leg of the trip to the total

total += map[pos][temp];

//mark that element in the row as invalid

map[pos][temp] = 0;

//mark that city as visited

vis[temp] = 1;

//set pos equal to temp since it is the new position

pos = temp;

//increment temp - needs to happen in both cases even if the initial weight was not zero

temp++;

}

//else, increment temp and visit the other necessary cities along the way since

//a city/current city has already been visited

else

{

temp++;

}

//

}

std::cout << "\nTotal distance = " << total;

}

Notice that while you do not traverse the same stretch of highway more than once in the same direction, it is possible to go back the other way. Add a step after 7 where the element representing the return distance is also invalidated.

Sample Run

City List

----------------

0) Lansing

1) Flint

2) Detroit

3) Jackson

4) Grand Rapids

5) Battle Creek

6) Kalamazoo

7) Clare

Enter start city: 7

Enter stop city: 5

Clare to Lansing = 85

Lansing to Flint = 53

Flint to Detroit = 77

Detroit to Lansing = 80

Lansing to Jackson = 34

Jackson to Detroit = 68

Can't get there from here

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

\* Lab27.cpp

\* Written by Twymun K. Safford

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

#include <iostream>

#include <string>

using namespace std;

#include "Lab27.h2"

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

\* main()

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

void main()

{

//start - starting city from user selection

//stop - end city from user selection

//pos- current position

// i - counter for city list

//temp - second index counter to seatch through all elements of 2d array for cities

int pos, temp, start, stop;

int i;

//total distance travelled

int total = 0;

//print statement for the city list with index

std::cout << "City List\n--------------------" << endl;

for (i = 0; i < CITY\_MAX; i++)

{

std::cout << i << ") " << city[i] << endl;

}

//ask user to input start city

std::cout << "\nEnter the index of the starting city: ";

cin >> start;

//ask user to enter output city

std::cout << "Enter the index of the stop city: ";

cin >> stop;

//set pos equal to start and temp to 0

pos = start;

//integer array - determine if a city has been visited yet or not

int vis[CITY\_MAX] = { 0 };

// starting city is always visited

vis[start] = 1;

//set temp to 0

temp = 0;

//iterate until start does not equal the end

while (pos != stop && pos < CITY\_MAX)// && pos < CITY\_MAX)

{

//for (i = 0; i < CITY\_MAX; i++)

//{

// map[i][temp] != 0 as well as the destination city is not visited

if ((map[pos][temp] != 0) && (vis[temp] == 0))

{

std::cout << city[pos] << " to " << city[temp] << " = " << map[pos][temp] << endl;

//add the weight for each leg of the trip to the total

total += map[pos][temp];

//mark that element in the row as invalid

map[pos][temp] = 0;

//mark that city as visited

vis[temp] = 1;

//set pos equal to temp since it is the new position

pos = temp;

//increment temp - needs to happen in both cases even if the initial weight was not zero

temp++;

}

//criteria to get stuck in a city

else

//else if (map[pos][temp]==0 && vis[temp] == 0)

{

cout << "Looks like we got stuck here. Can't make it to " << city[stop] << endl;

return;

}

//else, increment temp and visit the other necessary cities along the way since

//a city/current city has already been visited

/\*else

{

temp++;

}\*/

}

std::cout << "\nTotal distance = " << total;

}

**Screenshot:**

**Text

Description automatically generated**

Now it is possible to get stuck in a city. Alter the program so that you do not visit the same city more than once. Hint: Invalidate an entire row or column.

Sample Run

City List

----------------

0) Lansing

1) Flint

2) Detroit

3) Jackson

4) Grand Rapids

5) Battle Creek

6) Kalamazoo

7) Clare

Enter start city: 7

Enter stop city: 5

Clare to Lansing = 85

Lansing to Flint = 53

Flint to Detroit = 77

Detroit to Jackson = 68

Jackson to Battle Creek = 30

Total = 313

**Screenshot:**

Text

Description automatically generated

More sample runs:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Final Code:

Lab27.cpp:

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

\* Lab27.cpp

\* Written by Twymun K. Safford

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

#include <iostream>

#include <string>

using namespace std;

#include "Lab27.h2"

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

\* main()

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

void main()

{

//start - starting city from user selection

//stop - end city from user selection

//pos- current position

// i - counter for city list

//temp - second index counter to seatch through all elements of 2d array for cities

int pos, temp, start, stop;

int i;

//total distance travelled

int total = 0;

//print statement for the city list with index

std::cout << "City List\n--------------------" << endl;

for (i = 0; i < CITY\_MAX; i++)

{

std::cout << i << ") " << city[i] << endl;

}

//ask user to input start city

std::cout << "\nEnter the index of the starting city: ";

cin >> start;

//ask user to enter output city

std::cout << "Enter the index of the stop city: ";

cin >> stop;

//set pos equal to start and temp to 0

pos = start;

//integer array - determine if a city has been visited yet or not

int vis[CITY\_MAX] = { 0 };

// starting city is always visited

vis[start] = 1;

//set temp to 0

temp = 0;

//iterate until start does not equal the end

while (pos != stop && pos < CITY\_MAX)// && pos < CITY\_MAX)

{

// map[i][temp] != 0 as well as the destination city is not visited

if ((map[pos][temp] != 0) && (vis[temp] == 0))

{

std::cout << city[pos] << " to " << city[temp] << " = " << map[pos][temp] << endl;

//add the weight for each leg of the trip to the total

total += map[pos][temp];

//mark that element in the row as invalid

map[pos][temp] = 0;

//mark that city as visited

vis[temp] = 1;

//set pos equal to temp since it is the new position

pos = temp;

//increment temp - needs to happen in both cases even if the initial weight was not zero

temp++;

}

//criteria to get stuck in a city

//else

//{

// cout << "Looks like we got stuck here. Can't make it to " << city[stop] << endl;

// return;

//}

//else, increment temp and visit the other necessary cities along the way since

//a city/current city has already been visited

else

{

temp++;

}

}

std::cout << "\nTotal distance = " << total;

}

Header file:

// Lab 27 Cities and Distances

#define CITY\_MAX 8

int map[CITY\_MAX][CITY\_MAX] =

{ { 0,53,80,34, 75,35, 0, 85 },

{ 53, 0,77, 0, 0, 0, 0, 94 },

{ 80,77, 0,68, 0, 0, 0, 0 },

{ 34, 0,68, 0, 0,30, 0, 0 },

{ 75, 0, 0, 0, 0, 0,59,109 },

{ 35, 0, 0,30, 0, 0,47, 0 },

{ 0, 0, 0, 0, 59,47, 0, 0 },

{ 85,94, 0, 0,109, 0, 0, 0 }

};

string city[CITY\_MAX] =

{ "Lansing","Flint","Detroit","Jackson","Grand Rapids","Battle Creek","Kalamazoo","Clare" };