**Program Requirements**

Working on finding the shortest path for an emergency vehicle using the shortest path algorithm. Where the program will read in the data files with distances and different routes and output the shortest path to get to the destination entered by the user.

**Program Inputs:**

* Location names
* Starting Destination
* Destination
* Different Routes and their distances as text files

**Program Outputs:**

* Shortest path between starting and destination.

**Test Cases:**

**TEST CASE 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start index** | **End index** | **Shortest path** | **Distance** | **Continue (Y/N)** |
| **0** | **5** | **5 <- 4 <- 0** | **9 miles** | **Y** |
| **0** | **7** | **7 <- 6 <- 5 <- 4 <- 0** | **16 miles** | **N** |

**TEST CASE 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start index** | **End index** | **Shortest path** | **Distance** | **Continue (Y/N)** |
| **0** | **3** | **3 <- 0** | **5 miles** | **N** |

**TSET CASE 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Start index** | **End index** | **Shortest path** | **Distance** | **Continue (Y/N)** |
| **0** | **4** | **4 <- 0** | **7 miles** | **Y** |
| **0** | **2** | **2 <- 1 <- 0** | **3 miles** | **N** |

**Algorithm:**

1. **readBuildings Function**: reads the name of the buildings from the text file.

* Open file with the names of all the locations.
* Reads each line from the file and adds it to the vector of strings.
* Close the file.

1. **readMatrix Function:** reads the 2d adjacency matrix from the text file

* open the file with the 2d matrix.
* Have a loop to read all distances from the file and add it to the vector.
* Replace all -1 with 0 in the matrix to find distance for shortest path.
* Use a temporary vector row to add the distance into 2d vector.
* Close the file.

1. **printLocations Function:**  prints the index and their corresponding location names for the user.

* Have a loop to iterate over each value and print it to the user.

1. **printMatrix Function:** prints the adjacency matrix for the user.

* Using loops, printing the row and columns of the matrix.
* Print the current distance from the position.

1. **minDistance Function:**  finds the minimum distance vertex and it’s index.

* Have vectors with distances and checked vertices.
* Initialize minimum index and minimum.
* Go over each value to see if vertex and checked and if the distance is less than the current distance.
* Update the minimum index and the minimum.
* Return the minimum index to the user.

1. **dijkstrasAlgorithm Function:** finds the shortest path from the hospital and prints it for the user

* have vector for checked vertex, and distances with the predecessors.
* Have the starting point as the hospital.
* Find the minimum distance for each vertex and update it as a checked vertex.
* Update the predecessors and distances if necessary.
* print the shortest path from the hospital to the location for user with distance and predecessor.

1. **Main Function:** the front end for user interaction

* Read the names of the locations.
* Read the matrix from the text file.
* Print the matrix for user checking.
* Make the starting index as the hospital.
* Print the locations for the user to see.
* Ask the user to input destination of choice.
* Check for valid input by user.
* Call the algorithm to find the shortest path for user.
* As the user if they wish to continue.

**Flowchart:**

 

 

 

