MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 3: Directed Graph of Warehouse System

This assignment aims to help you practice **graph data structure and basic graph operations**. Your main task in this assignment is to develop a graph of Warehouse delivery routes using the **C programming language**. Please note that you need to implement the graph as an **adjacency list** in this assignment.

Overview:

A company in Cyprus seeks your assistance in establishing a warehouse network. Before setting up the network, they need to determine the routes between cities and towns that will facilitate deliveries between warehouses. Your task is to create an application to analyze potential routes from a provided file and deliver insights based on the data.

In this assignment, you are required to create a directed graph representing the warehouse route system based on the provided warehouse and route data. The warehouse locations will be provided in a file named `WarehouseLocations.txt`, and the route information will be provided in a file named `WarehouseRoutes.txt`. **Each warehouse will be represented as a vertex, and each route will be represented as a directed edge**. The weight of each edge will indicate the distance between warehouses in kilometers.

Internal Processing:

1. Read Data:

- Read Warehouse locations from `WarehouseLocations.txt` and route information from `WarehouseRoutes.txt`.
- Create a vertex for each warehouse
- Create directed edges for routes between warehouses.

2. Create Vertex Function:

- Create a function that takes graph and warehouse location as input.
- Create a new vertex for a warehouse location and add it to the graph.
- Please note that each vertex should store the warehouse location.

3. Create Edge Function:

- Create a function that takes the graph, departure, destination and distance between departure and destination locations.
- Create a directed edge from departure to destination with the weight equal to distance.
- Update indegree for destination and outdegree for departure

4. Print Graph Function:

- Print the adjacency list of the resulting graph, showing the vertices and directed edges.
- Include distance between departure and destination as well.

5. Generate Stats:

- Print the Warehouse that receives deliveries from the most different locations (indegree), the warehouse that sends delivery to the most different locations (outdegree), the route with highest

distance, the route with smallest distance.
- If there is more than one route with the same distance, you should **print all of them**.
- If there is more than one warehouse that receives from most different locations or delivers to most different locations, you should **print all of them.**
- Don't forget to print departure and destination locations as well. (Check sample output!)

6. Check Distance:

- Take departure and destination warehouse locations from the command line.
- Check if a path is available from departure location to destination location.
    - Which means check if you can go from departure to destination.
- If there is an available path, print it with other warehouse locations that you pass through, show distance between each location and provide total distance at the end of data.

Input:

The program will have two data files as inputs, and additional two command line arguments: (1) deperature warehouse and (2) destination warehouse:
- `WarehouseLocations.txt` with warehouse locations information.
- `WarehouseRoutes.txt` with available routes between warehouses.

Data Format:

`WarehouseLocations.txt`:
```
Guzelyurt
Lefkosa
Girne
Gazimagusa
...
```

`WarehouseRoutes.txt`:
```
Departure_Location;Distance;Destination_Location,Distance,Destionation_Location...
Lefkosa;35;Haspolat;10;Gonyeli;60;Gazimagusa;45;Guzelyurt…
Guzelyurt;15;Lefke;20;Iskele…
…
```

Term	Definition
Departure_Location	The Location where the delivery starts
Distance	Distance between Departure_Location and Delivery_Location
Destination_Location	The Location where the delivery ends

Functions:

In this assignment, you are given a C file called "template2.c" attached. This will be the template code which you need to **strictly follow**. Inside the "template2.c" file, all function prototypes are declared, the main function is already written along with a few others. You will basically need to fill in the blanks and write the code for the following functions:

- **struct graphHead * createGraph(void):**
  Creates the graph. (Already Given)
- **void insertVertex(struct graphHead *head, char *data):**
  Creates a vertex for a warehouse location (Already Given)
- **int insertArc(struct graphHead *head, char *fromKey, char *toKey ,int weight):**
  Creates a directed edge (route) between destination (fromKey) and departure (toKey) locations with

distance as weight of the arc.

- **void printAllArcOfVertex(struct graphVertex *graphVertexName):**
  Prints all arcs of a vertex. (Prints all routes a warehouse location delivers to)
- **void printAllVertex(struct graphHead *graphName):**
  Prints all vertex data. (Prints all warehouse locations.).
- **void printAllVertexWithRoutes(struct graphHead *graphName):**
  Prints vertices which have a delivery route. Use **printAllArcOfVertex** here.
- **void readRoutes(char *fileName, struct graphHead *warehouseGraph):**
  Reads route data from a file.
- **void readWarehouseNames(char *fileName, struct graphHead *warehouseGraph):**
  Reads warehouse locations from a file.
- **void getMostDeliveries(struct graphHead *warehouseGraph):**
  Finds the warehouses which get most deliveries from other warehouses.
- **void sendMostDeliveries(struct graphHead *warehouseGraph):**
  Finds the warehouses which send most deliveries to other warehouses.
- **void routesWithHighestDistance(struct graphHead *warehouseGraph):**
  Finds all the routes (arcs or edges) with the highest distance.
- **void routesWithLowestDistance(struct graphHead *warehouseGraph):**
  Finds all the routes (arcs or edges) with the lowest distance.
- **int findPath(struct graphVertex *currentVertex, struct graphVertex *destinationVertex, int *totalDistance):**
  In this function you should create an algorithm to find if a path exists between 2 given warehouse locations, if a path exists, you should print it with their distance between each location. Check sample output.
- **void checkDistance(struct graphHead *warehouseGraph, char *departure, char *destination):**
  In this function, it checks if departure and destination warehouse locations exist. If they exist, call **findPath** to print the path and calculate total distance to print as well.

Important Notes for Implementation

- The template code assumes the data files you are going to read are called "WarehouseLocations.txt" and "WarehouseRoutes.txt" and that they are stored directly in the code's immediate directory. Note that if you are using an IDE like Clion, you will need to store the file in the "cmake-build-debug" folder.
- **You are not allowed to change any code in the template**, and you should abide by the prototypes and structures given. **Changing anything in the template may result in an immediate 0 (zero).** The places where you need to write your code are marked with the comment: "WRITE YOUR CODE HERE".
- You need to ensure usage of comments to clarify your codes.
- You can add helper functions.
- You must make use of insertVertex in readWarehouseNames function, while also making use of insertArc in readRoutes function.

Output:

A sample output for four warehouse locations has been given below. Also a sample output with sample warehouse locations and routes has been given as an appendix. You can use them to test your program.

```
TestLocations.txt has been opened successfully
TestRoutes.txt has been opened successfully

Warehouse and route data read successfully.
```

```
Read warehouse locations:
WarehouseA
WarehouseB
WarehouseC
WarehouseD

Read routes:
WarehouseA | -50-> WarehouseB | -35-> WarehouseC |
WarehouseB | -35-> WarehouseC |
WarehouseC | -25-> WarehouseA | -50-> WarehouseB | -24-> WarehouseD |

Warehouses that receives deliveries from the most different locations (2):
- WarehouseB
- WarehouseC

Warehouses that sends deliveries to most different locations (3):
- WarehouseC

Routes with the highest distance (50):
WarehouseA -50-> WarehouseB
WarehouseC -50-> WarehouseB

Routes with the smallest distance (24):
WarehouseC -24-> WarehouseD

Searching for a route from WarehouseA to WarehouseD...
WarehouseD <-24- WarehouseC <-35- WarehouseB <-50- WarehouseA
Total Distance: 109
```

Incremental and Modular Development

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions.

Professionalism and Ethics

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions. Specifically, you are expected to implement some helper functions.

Rules

- You need to write your program by using C programming.
- You need to name your file with your student id, e.g., 1234567.c, and submit it to ODTUCLASS.
- Code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading.
- **You need to strictly follow the specifications given in this document.**
- It is suggested to use helper functions to support modular development.

Grading Scheme

The assignment will be graded based on the following scheme:

Grading Item	**Score**
insertArc	5
printAllArcOfVertex	5
printAllVertex	5
printAllVertexWithRoutes	5
readRoutes	7.5
readWarehouseNames	7.5
getMostDeliveries	5
sendMostDeliveries	5
routesWithHighestDistance	10
routesWithLowestDistance	10
findPath	25
checkDistance	10

Please note that if your function does not produce an expected output, **you will not receive any mark from that function**. You need to ensure your code runs and is testable. **If you submit a code which does not compile, you will automatically get zero.** Please note that code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading. Therefore, **providing an expected output does not guarantee a full mark.**