CODE EXPLANATION:

Function DisplayRoute and FindRoutes:

- The function `findRoutes` is a recursive depth-first search (DFS) algorithm that explores all possible paths from a source station (`src`) to a destination station (`dest`).
- The function `displayRoute` is called whenever a valid route is found, printing the stations along the route and the total cost.
- The recursion ensures that all paths are explored by traversing through the network's edges, while the `visited` array ensures no station is visited more than once in a single path.
  Example:
  Input: find_routes B D
  Output: B -> C -> D, Cost : 13
  B->I->H->C->D, Cost : 20

Function Station Info:

-> Search for Station: It first calls `getStationIndex` to find the index of the station with the given name (`stationName`). If the station is not found (`stationIndex` is `-1`), it prints "Station not found." and exits.
-> Display Station Details: If the station is found, it prints the station's name.
-> List Lines: It then lists the metro lines serving the station by looping through the `lines` array in the station's data and printing each line name, separated by commas.
-> List Adjacent Stations: Finally, it prints all adjacent (connected) stations by iterating through a linked list of `Edge` structures, where each edge represents a connection to another station. For each adjacent station, it prints the station's name and the fare (cost) to travel to it. Commas are used to separate adjacent stations in the output.

Function Display Intersection :

- Purpose if function is to display stations that are part of multiple lines in the metro network.
- Loops through each station in the metro network.
- Checks if a station is connected to more than one line by verifying if line countfor that station is greater than 1.
- If a station serves multiple lines, it prints the station's nam

Display Terminal station:

function lists the terminal stations for each metro line in the network.

Purpose of this function is to display the starting and ending stations of each metro line.

Iterates over all lines in the Metro Network.

For each line, it prints the line's name, along with the names of its starting and ending stations.

This function provides a quick overview of the terminal points for each metro line.

Find Nearest Intersection :

- This uses a bfs, taking the network structure (the whole metro network) and the station index inside of the stations array (not the name)
- This station index is found in the main function using a for loop and strcmp
- As is procedure in bfs, we initialize a queue
- Make the arrays, visited, cost and parent to help with the bsf as is explained in the code, visited is initialized to 0 to make all nodes equal to not visited
- We make the first element of the queue equal to the stationIndex given i.e. start point of the bsf
- Using while loop we search for station which is the intersection station (because we are using bsf, we find shortest path)
- Then backtrack to the stationindex and print the path in reverse order to find the nearest intersection path
- If no path is found, i.e. if is never entered, while is exhausted and no intersection found statement is sent to stdout

OUTPUT:

Find Routes

```
Enter command (find_routes, station_info, display_intersections, display_terminal_stations, find_path_to_nearest_i
ntersection, or exit): find_routes
Enter source and destination stations: b
d
Possible routes:
b -> i -> h -> c -> d, Cost: 20
b -> c -> d, Cost: 13
```

Station Info

```
Enter command (find_routes, station_info, display_intersections, display_terminal_stations, find_path_to_nearest_i
ntersection, or exit): station_info
Enter station name: g
Station Name: g
Lines: greenline
Adjacent Stations: f (9), c (3)
```

Display Intersections

```
Enter command (find_routes, station_info, display_intersections, display_terminal_stations, find_path_to_nearest
_intersection, or exit): display_intersections
b, 2
c, 2
i, 2
```

Display Terminal Stations

```
Enter command (find_routes, station_info, display_intersections, display_terminal_stations, find_path_to_nearest_i
ntersection, or exit): display_terminal_stations
redline: a, d
greenline: e, k
blueline: b, l
```

Find Path to Nearest Intersection

```
Enter command (find_routes, station_info, display_intersections, display_terminal_stations, find_path_to_nearest_i
ntersection, or exit): find_path_to_nearest_intersection
Enter station name: f
Path to nearest intersection: f -> g -> c, Cost: 12
```