

NAME : PREM SAH
PRN : 123B1B234
DIV : D

ASSIGNMENT NO.5

Implement a navigation system for a delivery service using a circular linked list to represent routes. The navigation system should support the following functionalities:\

- a. Add route
- b. Remove route
- c. Display route

CODE :

```
#include <iostream>
#include <string>
using namespace std;

// Node structure for each route in the circular linked list
class RouteNode {
public:
    string routeName;
    RouteNode* next;

    RouteNode(string name) : routeName(name), next(nullptr) {}
};

// NavigationSystem class using circular linked list
class NavigationSystem {
private:
    RouteNode* head;

public:
    NavigationSystem() : head(nullptr) {}

    // Add route to the circular linked list
    void addRoute(const string& routeName) {
        RouteNode* newRoute = new RouteNode(routeName);

        if (!head) {
            head = newRoute;
            head->next = head; // Points to itself to create the circular structure
        } else {
            RouteNode* temp = head;
            while (temp->next != head) {
```

```

        temp = temp->next;
    }
    temp->next = newRoute;
    newRoute->next = head; // Completes the circle
}
cout << "Route added: " << routeName << endl;
}

```

```

// Remove a route from the circular linked list
void removeRoute(const string& routeName) {
    if (!head) {
        cout << "No routes to remove!" << endl;
        return;
    }

```

```

    RouteNode* temp = head;
    RouteNode* prev = nullptr;

```

```

// If the route to remove is the head
if (temp->routeName == routeName) {
    if (temp->next == head) { // Only one route exists
        delete head;
        head = nullptr;
    } else {
        RouteNode* last = head;
        while (last->next != head) {
            last = last->next;
        }
        head = head->next;
        last->next = head;
        delete temp;
    }
    cout << "Route removed: " << routeName << endl;
    return;
}

```

```

// Search for the route to remove
do {
    prev = temp;
    temp = temp->next;
} while (temp != head && temp->routeName != routeName);

```

```

// If the route was not found
if (temp == head) {

```

```

        cout << "Route not found!" << endl;
        return;
    }

    // Unlink the node and delete it
    prev->next = temp->next;
    delete temp;
    cout << "Route removed: " << routeName << endl;
}

// Display all routes in the circular linked list
void displayRoutes() {
    if (!head) {
        cout << "No routes available!" << endl;
        return;
    }

    RouteNode* temp = head;
    cout << "Routes: ";
    do {
        cout << temp->routeName << " -> ";
        temp = temp->next;
    } while (temp != head);
    cout << "(back to start)" << endl;
}

};

int main() {
    NavigationSystem navSystem;

    // Add routes
    navSystem.addRoute("Route A");
    navSystem.addRoute("Route B");
    navSystem.addRoute("Route C");

    // Display routes
    cout << "\nCurrent Routes:" << endl;
    navSystem.displayRoutes();

    // Remove a route
    cout << "\nRemoving Route B:" << endl;
    navSystem.removeRoute("Route B");

    // Display updated routes

```

```
cout << "\nUpdated Routes:" << endl;  
navSystem.displayRoutes();  
  
return 0;  
}
```

Output :

**Route added: Route A
Route added: Route B
Route added: Route C**

**Current Routes:
Routes: Route A -> Route B -> Route C -> (back to start)**

**Removing Route B:
Route removed: Route B**

**Updated Routes:
Routes: Route A -> Route C -> (back to start)**

main.cpp

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node structure for each route in the circular linked list
6 class RouteNode {
7 public:
8     string routeName;
9     RouteNode* next;
10
11     RouteNode(string name) : routeName(name), next(nullptr) {}
12 };
13
14 // NavigationSystem class using circular linked list
15 class NavigationSystem {
16 private:
17     RouteNode* head;
18
19 public:
20     NavigationSystem() : head(nullptr) {}
21
22     // Add route to the circular linked list
23 void addRoute(const string& routeName) {
24     RouteNode* newRoute = new RouteNode(routeName);
25
26     if (!head) {
27         head = newRoute;
28         head->next = head; // Points to itself to create the circular structure
29     } else {
30         RouteNode* temp = head;
31         while (temp->next != head) {
32             temp = temp->next;
33         }
34         temp->next = newRoute;
35         newRoute->next = head; // Completes the circle
36     }
37     cout << "Route added: " << routeName << endl;
38 }
39
40 // Remove a route from the circular linked list
41 void removeRoute(const string& routeName) {
42     if (!head) {
43         cout << "No routes to remove!" << endl;
44         return;
45     }
```

main.cpp

```
46
47 RouteNode* temp = head;
48 RouteNode* prev = nullptr;
49
50 // If the route to remove is the head
51 if (temp->routeName == routeName) {
52     if (temp->next == head) { // Only one route exists
53         delete head;
54         head = nullptr;
55     } else {
56         RouteNode* last = head;
57         while (last->next != head) {
58             last = last->next;
59         }
60         head = head->next;
61         last->next = head;
62         delete temp;
63     }
64     cout << "Route removed: " << routeName << endl;
65     return;
66 }
67
68 // Search for the route to remove
69 do {
70     prev = temp;
71     temp = temp->next;
72 } while (temp != head && temp->routeName != routeName);
73
74 // If the route was not found
75 if (temp == head) {
76     cout << "Route not found!" << endl;
77     return;
78 }
79
80 // Unlink the node and delete it
81 prev->next = temp->next;
82 delete temp;
83 cout << "Route removed: " << routeName << endl;
84 }
85
86 // Display all routes in the circular linked list
87 void displayRoutes() {
88     if (!head) {
89         cout << "No routes available!" << endl;
90         return;
```

```

91     }
92
93     RouteNode* temp = head;
94     cout << "Routes: ";
95     do {
96         cout << temp->routeName << " -> ";
97         temp = temp->next;
98     } while (temp != head);
99     cout << "(back to start)" << endl;
100 }
101 };
102
103 int main() {
104     NavigationSystem navSystem;
105
106     // Add routes
107     navSystem.addRoute("Route A");
108     navSystem.addRoute("Route B");
109     navSystem.addRoute("Route C");
110
111     // Display routes
112     cout << "\nCurrent Routes:" << endl;
113     navSystem.displayRoutes();
114
115     // Remove a route
116     cout << "\nRemoving Route B:" << endl;
117     navSystem.removeRoute("Route B");
118
119     // Display updated routes
120     cout << "\nUpdated Routes:" << endl;
121     navSystem.displayRoutes();
122
123     return 0;
124 }
125

```

Output

```

/tmp/oDU39ovybD.o
Route added: Route A
Route added: Route B
Route added: Route C

Current Routes:
Routes: Route A -> Route B -> Route C -> (back to start)

Removing Route B:
Route removed: Route B

Updated Routes:
Routes: Route A |-> Route C -> (back to start)

=== Code Execution Successful ===

```