NAME : PREM SAH
PRN : 123B1B234
DIV : D

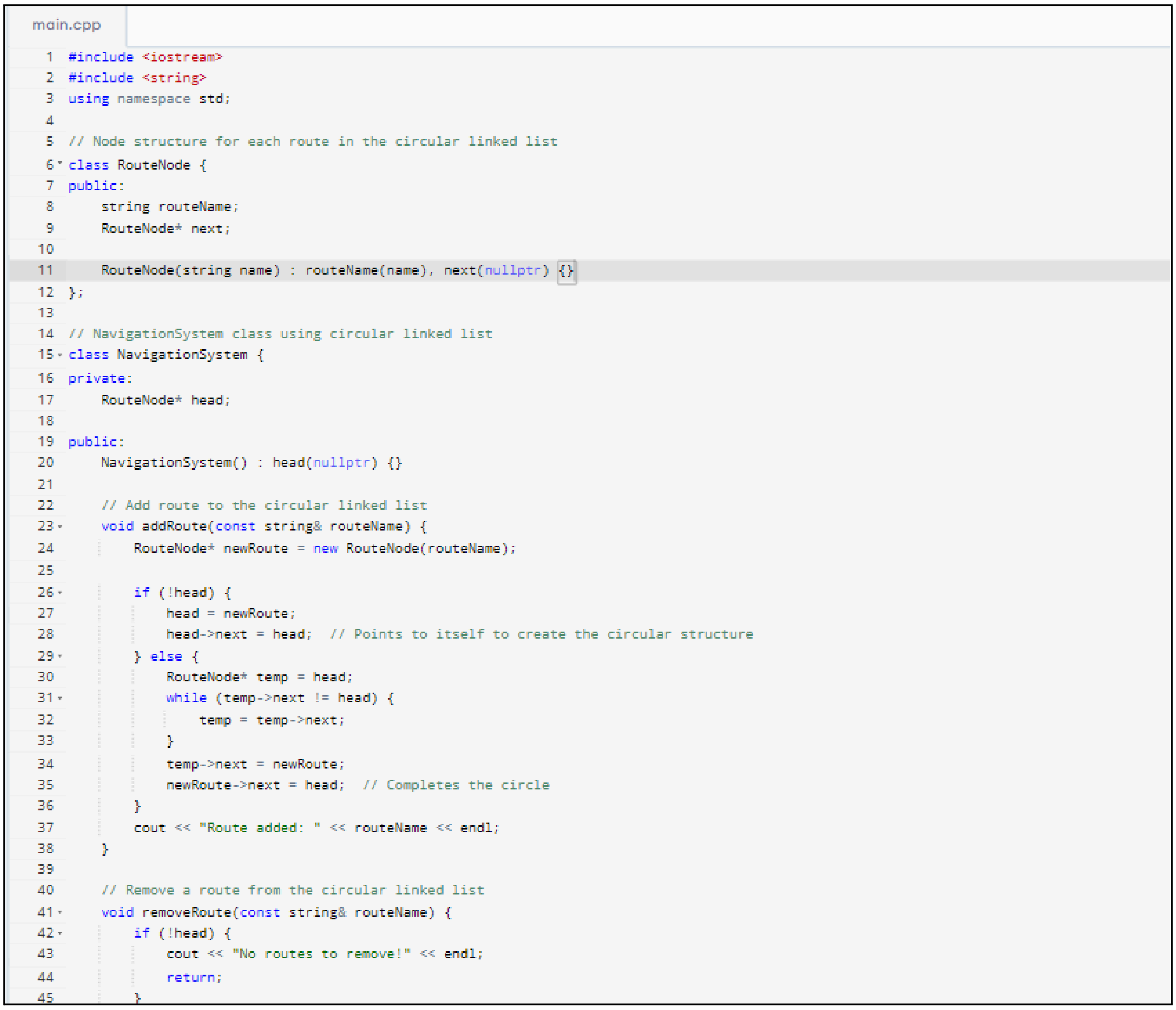
ASSIGNMENT NO.5

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

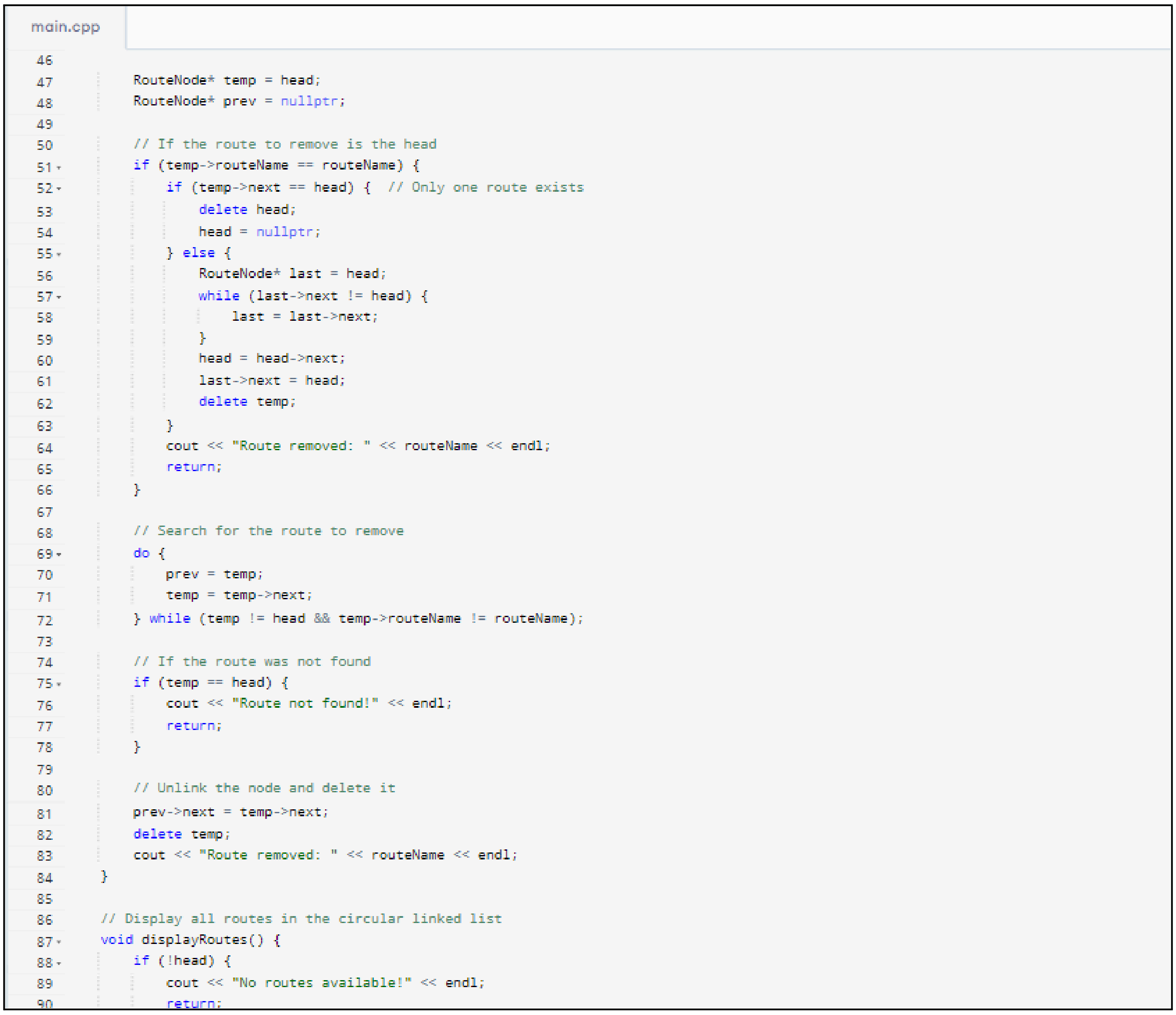
CODE :
```
#include <iostream>
#include <string>
using namespace std;

// 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 :

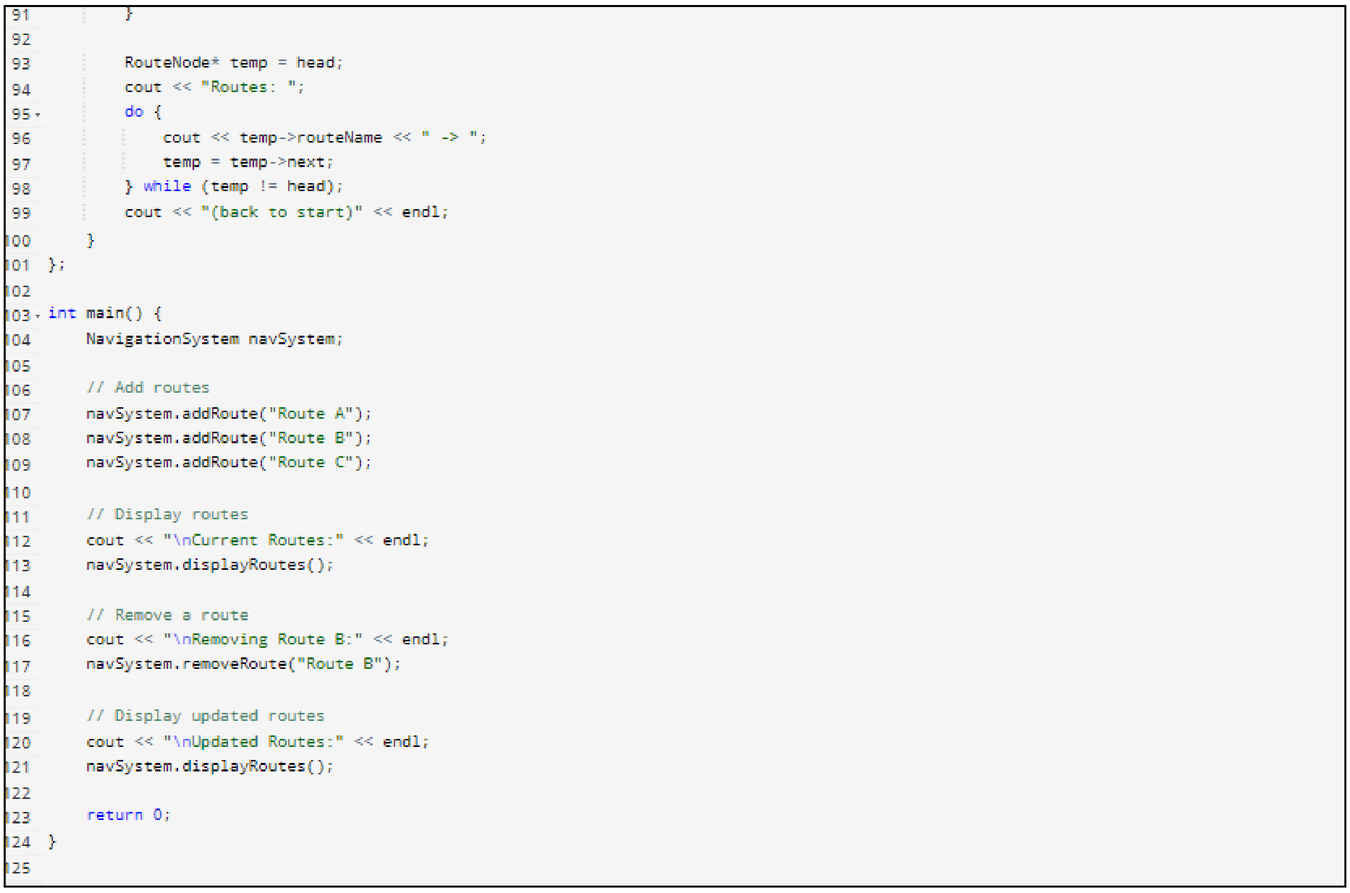
```
Route added: Route A
Route added: Route B
Route added: Route C

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

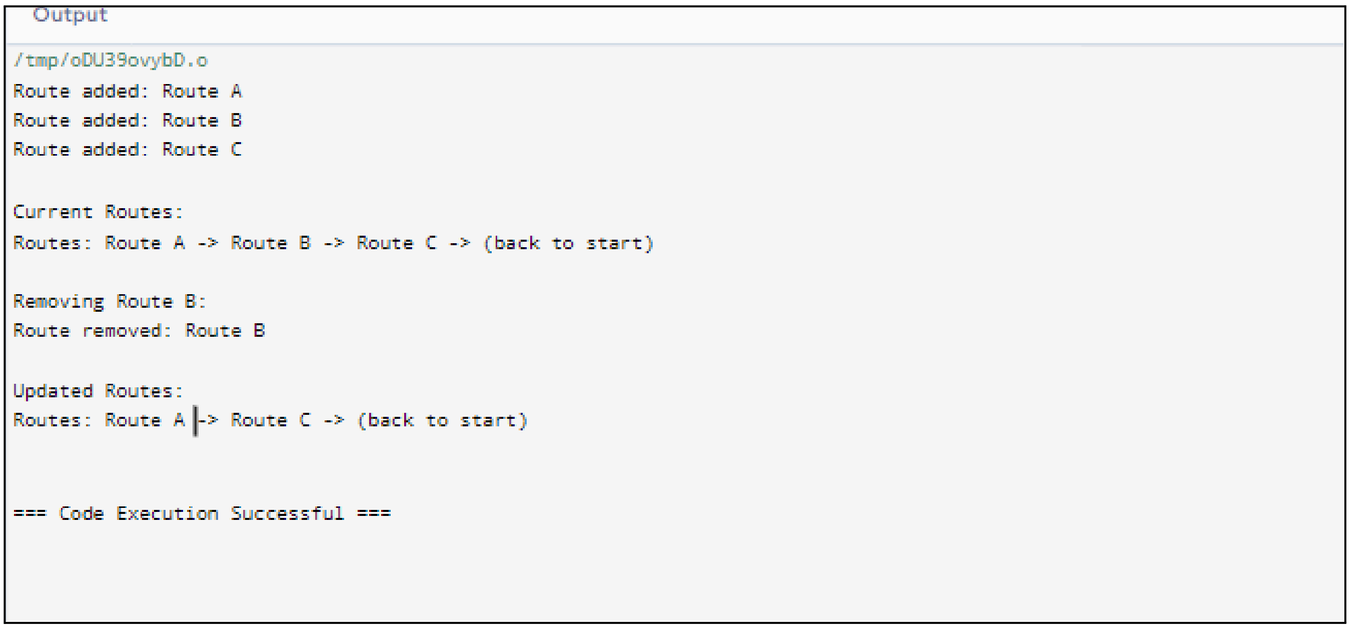
```cpp
#include <iostream>
#include <string>
using namespace std;

// 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;
        }
```



```cpp

        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;
```



```cpp
        }

        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

```
/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 ===
```