PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

S.Y. B. TECH **Year:** 2024 – 25 **Semester:** I

Name: Sonawane Prachi Mahendra. **PRN:** 124B2B018

Department : Computer Engineering **Division:** B

Course : Data Structures Laboratory **Course Code:** BCE23PC02

Date: 11/09/24

Assignment – 5

- **Aim:**

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

  1. Add route
  2. Remove route
  3. Display route

- **Source Code:**

```
#include <iostream> #include

<string>

class Node

{

   public: std::string route;

   Node* next;

   Node(const std::string route)

   {
```

```
    next=NULL; route=route;
    }
};
class CircularLinkedList
{
    private:
    Node* head;
    public:
    CircularLinkedList()
    {
        head=NULL;
    }
// Function to add a route
    void addRoute(const std::string& route)
    {
        Node* newNode = new Node(route); if
        (!head) {
            head = newNode;
            newNode->next = head; // Point to itself
        }
        else
        {
            Node* temp = head;
            while (temp->next != head)
```

```
        {
        temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = head;
    }
    std::cout << "Route added: " << route << std::endl;
}
void removeRoute(const std::string& route)
{
    if (!head)
    {
    std::cout << "No routes to remove." << std::endl; return;
    }
    Node* current = head; Node*
    previous = nullptr; do
    {
            if (current->route == route) { if
            (previous) {
                previous->next = current->next;
            }
            else {
                Node* temp = head;
```

```
            while (temp->next != head) { temp
            = temp->next;
        }
        temp->next = head->next; head =
        head->next;
    }
    delete current;
    std::cout << "Route removed: " << route << std::endl; return;
  }
  previous = current; current =
  current->next;
 } while (current != head);
 std::cout << "Route not found: " << route << std::endl;
}
 void displayRoutes() { if
 (!head) {
  std::cout << "No routes available." << std::endl; return;
 }
 Node* current = head; std::cout
 << "Routes: "; do {
  std::cout << current->route << " "; current
  = current->next;
```

```
        } while (current != head); std::cout
        << std::endl;

    }
};
int main()
{
    CircularLinkedList routes; int
    choice;
    std::string route;
    do {
        std::cout << "\n1. Add Route\n2. Remove Route\n3. Display Routes\n4. Exit\n";
        std::cout << "Enter your choice: ";
        std::cin >> choice;
            switch (choice) { case
            1:

            std::cout << "Enter route: ";
            std::cin.ignore(); std::getline(std::cin,
            route); routes.addRoute(route); break;

            case 2:
            std::cout << "Enter route to remove: ";
            std::cin.ignore();
```

```
            std::getline(std::cin,                route);
            routes.removeRoute(route); break;

            case 3:
            routes.displayRoutes(); break;
            case 4:
            std::cout << "Exiting..." << std::endl; break;

            default:
            std::cout << "Invalid choice. Please try again." << std::endl;
        }
    } while (choice != 4); return 0;
}
```

- **Screen shots of Output:**

1.

Output

```
1. Add Route
2. Remove Route
3. Display Routes
4. Exit
Enter your choice: 1
Enter route: hello
Route added: hello

1. Add Route
2. Remove Route
3. Display Routes
4. Exit
Enter your choice: 3
Routes: hello

1. Add Route
2. Remove Route
3. Display Routes
4. Exit
Enter your choice: 2
Enter route to remove: hello
Route removed: hello

1. Add Route
2. Remove Route
3. Display Routes
4. Exit
Enter your choice: 4
Exiting...
```

- **Conclusion:**

  Hence, we studied about Circular linked list and its operations like insertion, deletion, traversing, etc.