Tabish Ahmad

23103145

Group :- G3 [3B]

Assignment - 8

1. Write a program to implement a circular linked list and its basic operations like

a. insertion,

I. at the beginning

II. at the end

III. at any specific location

b. deletion,

I. at the beginning

II. at the end

```
void insertAtBeginning(int data) {
    Node* newNode = new Node{data, nullptr};
    if (!head) {
        head = newNode;
        newNode->next = head;
    } else {
        Node* temp = head;
        while (temp->next != head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = head;
        head = newNode;
```

III. at any specific location

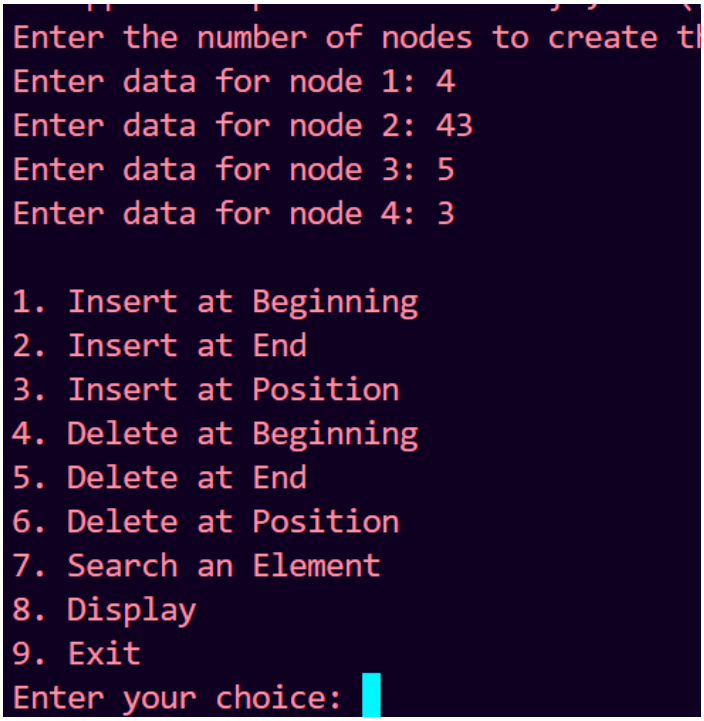
c. Searching an element

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
class CircularLinkedList {
    Node* head;
public:
    CircularLinkedList() {
        head = nullptr;}
    void createList(int n) {
        if (n <= 0) return;
        for (int i = 0; i < n; i++) {
            int data;
            cout << "Enter data for node " << i + 1 << ": ";
            cin >> data;
            insertAtEnd(data);
        }}


        }}
    void insertAtEnd(int data) {
        Node* newNode = new Node{data, nullptr};
        if (!head) {
            head = newNode;
            newNode->next = head;
        } else {
            Node* temp = head;
            while (temp->next != head) {
                temp = temp->next;
            }
            temp->next = newNode;
```

```
        newNode->next = head;
    }}
  void insertAtPosition(int data, int position) {
    if (position == 0) {
      insertAtBeginning(data);
      return;}
    Node* newNode = new Node{data, nullptr};
    Node* temp = head;
    for (int i = 0; i < position - 1 && temp->next !=
head; i++) {
      temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
    }
  void deleteAtBeginning() {
    if (!head) return;
    Node* temp = head;
    if (temp->next == head) {
      delete head;
      head = nullptr;
    } else {
      Node* last = head;
      while (last->next != head) {
        last = last->next;
      }
      head = head->next;
      last->next = head;
      delete temp;
    }}
  void deleteAtEnd() {
    if (!head) return;
    Node* temp = head;
    if (temp->next == head) {
      delete head;
      head = nullptr;
    } else {
      Node* prev = nullptr;
      while (temp->next != head) {
        prev = temp;
        temp = temp->next;   }
      prev->next = head;
      delete temp;
    }}
  void deleteAtPosition(int position) {
    if (position == 0) {
      deleteAtBeginning();
      return;}
    Node* temp = head;
    Node* prev = nullptr;
    for (int i = 0; i < position && temp->next != head;
i++) {
      prev = temp;
      temp = temp->next;}
    if (prev) {
      prev->next = temp->next;
      delete temp;
    }}
  bool search(int key) {
    if (!head) return false;
    Node* temp = head;
    do {
      if (temp->data == key) return true;
      temp = temp->next;
    } while (temp != head);
    return false;}
  void display() {
    if (!head) return;
```

```
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;  }
};
int main() {
    CircularLinkedList cll;
    int n, choice, data, position;
    cout << "Enter the number of nodes to create the
circular linked list: ";
    cin >> n;
    cll.createList(n);
    while (true) {
        cout << "\n1. Insert at Beginning\n2. Insert at
End\n3. Insert at Position\n4. Delete at Beginning\n5.
Delete at End\n6. Delete at Position\n7. Search an
Element\n8. Display\n9. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Enter data: ";
                cin >> data;
                cll.insertAtBeginning(data);
                cll.display();
                break;
            case 2:
                cout << "Enter data: ";
                cin >> data;
                cll.insertAtEnd(data);
                cll.display();
                break;
            case 3:
                cout << "Enter data and position: ";
                cin >> data >> position;
                cll.insertAtPosition(data, position);
                cll.display();
                break;
            case 4:
                cll.deleteAtBeginning();
                cll.display();
                break;
            case 5:
                cll.deleteAtEnd();
                cll.display();
                break;
            case 6:
                cout << "Enter position: ";
                cin >> position;
                cll.deleteAtPosition(position);
                cll.display();
                break;
            case 7:
                cout << "Enter element to search: ";
                cin >> data;
                cout << (cll.search(data) ? "Element found." :
"Element not found.") << endl;
                cll.display();
                break;
            case 9:
                return 0;
            default:
                cout << "Invalid choice." << endl;    }
    }}
```



2. Write a program to reverse a circular linked list.

```
#include <iostream>
using namespace std;
struct Node {
   int data;
   Node* next;
};
class CircularLinkedList {
   Node* head;
public:
   CircularLinkedList() {
      head = nullptr;
   }
   void createList(int n) {
      if (n <= 0) return;
      for (int i = 0; i < n; i++) {
         int data;
         cout << "Enter data for node " << i + 1 << ": ";
         cin >> data;
         insertAtEnd(data);
      }}
   void insertAtEnd(int data) {
      Node* newNode = new Node{data, nullptr};
      if (!head) {
         head = newNode;
         newNode->next = head;
      } else {
         Node* temp = head;
         while (temp->next != head) {
            temp = temp->next;
         }
         temp->next = newNode;
         newNode->next = head;
      }}
   void reverse() {
      if (!head || head->next == head) return;
      Node *prev = head, *current = head->next, *next
= nullptr;
      while(current!=head){
         next = current->next;
         current->next = prev;
         prev = current;
         current = next;
      }
      head->next = prev;
      head= prev;
   }
   void display() {
      if (!head) return;
      Node* temp = head;
      do {
         cout << temp->data << " ";
         temp = temp->next;
      } while (temp != head);
      cout << endl;}
};
```

```
int main() {
   CircularLinkedList cll;
   int n;
   cout << "Enter the number of nodes to create the circular linked list: ";
   cin >> n;
   cll.createList(n);
   cout << "Original Circular Linked List: ";
   cll.display();
   cll.reverse();
   cout << "Reversed Circular Linked List: ";
   cll.display();
   return 0;}
```

```
Enter the number of nodes to create
3
Enter data for node 1: 4
Enter data for node 2: 5
Enter data for node 3: 6
Original Circular Linked List: 4 5
Reversed Circular Linked List: 6 5
```

3. Write a program to count the number of elements of Circular Linked List

```
#include <iostream>
using namespace std;
struct Node {
   int data;
   Node* next;
};
class CircularLinkedList {
   Node* head;
public:
   CircularLinkedList() {
      head = nullptr;}
   void createList(int n) {
      if (n <= 0) return;
      for (int i = 0; i < n; i++) {
         int data;
         cout << "Enter data for node " << i + 1 << ": ";
         cin >> data;
         insertAtEnd(data);
      }}
   void insertAtEnd(int data) {
      Node* newNode = new Node{data, nullptr};
      if (!head) {
         head = newNode;
         newNode->next = head;
      } else {
         Node* temp = head;
         while (temp->next != head) {
            temp = temp->next;
         }
         temp->next = newNode;
         newNode->next = head;
      }}
   int countNodes() {
      if (!head) return 0;
      int count = 0;
      Node* temp = head;
      do {
         count++;
         temp = temp->next;
      } while (temp != head);
      return count;}
   void display() {
      if (!head) return;
      Node* temp = head;
      do {
         cout << temp->data << " ";
         temp = temp->next;
      } while (temp != head);
```

```
    cout << endl;}
};
int main() {
    CircularLinkedList cll;
    int n;
    cout << "Enter the number of nodes to create the circular linked list: ";
    cin >> n;
    cll.createList(n);
    cout << "Circular Linked List: ";
    cll.display();
    int count = cll.countNodes();
    cout << "Number of elements in the Circular Linked List: " << count << endl;
    return 0;
}
```

```
Enter the number of nodes to create t
Enter data for node 1: 4
Enter data for node 2: 5
Enter data for node 3: 8
Circular Linked List: 4 5 8
Number of elements in the Circular Li
```

4). Write a program to sort the elements Circular linked list.

```
#include <iostream>
using namespace std;
struct Node {
    float data;
    Node* next;
};
class CircularLinkedList {
    Node* head;
public:
    CircularLinkedList() {
        head = nullptr;
    }
    void createList(int n) {
        if (n <= 0) return;
        for (int i = 0; i < n; i++) {
            float data;
            cout << "Enter data for node " << i + 1 << ": ";
            cin >> data;
            insertAtEnd(data);
        }}
    void insertAtEnd(float data) {
        Node* newNode = new Node{data, nullptr};
        if (!head) {
            head = newNode;
            newNode->next = head;
        } else {
            Node* temp = head;
            while (temp->next != head) {
                temp = temp->next;
            }
            temp->next = newNode;
            newNode->next = head;
        }}
    void sortList() {
        if (!head || head->next == head) return;
        Node* current = head;
        Node* index = nullptr;
        int temp;
        do {
            index = current->next;
            while (index != head) {
                if (current->data > index->data) {
                    temp = current->data;
                    current->data = index->data;
                    index->data = temp;
```

```
            }
            index = index->next;
        }
        current = current->next;
    } while (current->next != head);}
  void display() {
    if (!head) return;
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
  }
};
int main() {
  CircularLinkedList cll;
  int n;
  cout << "Enter the number of nodes to create the
circular linked list: ";
  cin >> n;
  cll.createList(n);
  cout << "Original Circular Linked List: ";
  cll.display();
  cll.sortList();
  cout << "Sorted Circular Linked List: ";
  cll.display();
  return 0;
}
```

```
Enter the number of nodes to create the circular linked list: 5
Enter data for node 1: 4
Enter data for node 2: 8
Enter data for node 3: 9
Enter data for node 4: 88
Enter data for node 5: 22
Original Circular Linked List: 4 8 9 88 22
Sorted Circular Linked List: 4 8 9 22 88
```

5. Write a program to Print and remove duplicate elements from a Circular Linked List.

```
#include <iostream>
using namespace std;
struct Node {
  int data;
  Node* next;
};
class CircularLinkedList {
  Node* head;
public:
  CircularLinkedList() {
    head = nullptr;
  }
  void createList(int n) {
    if (n <= 0) return;
    for (int i = 0; i < n; i++) {
      int data;
      cout << "Enter data for node " << i + 1 << ": ";
      cin >> data;
      insertAtEnd(data);
    }}
  void insertAtEnd(int data) {
    Node* newNode = new Node{data, nullptr};
    if (!head) {
      head = newNode;
      newNode->next = head;
    } else {
```

```
        Node* temp = head;
        while (temp->next != head) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = head;}}
    void removeDuplicates() {
        if (!head || head->next == head) return;
        Node* current = head;
        do {
            Node* temp = current;
            while (temp->next != head) {
                if (current->data == temp->next->data) {
                    Node* duplicate = temp->next;
                    temp->next = duplicate->next;
                    delete duplicate;
                } else {
                    temp = temp->next;
                }
            }
            current = current->next;
        } while (current->next != head);
    }
    void display() {
        if (!head) return;
        Node* temp = head;
        do {
            cout << temp->data << " ";
            temp = temp->next;
        } while (temp != head);
        cout << endl;
    }
};
int main() {
    CircularLinkedList cll;
    int n;
    cout << "Enter the number of nodes to create the
circular linked list: ";
    cin >> n;
    cll.createList(n);
    cout << "Original Circular Linked List: ";
    cll.display();
    cll.removeDuplicates();
    cout << "Circular Linked List after removing
duplicates: ";
    cll.display();
    return 0;
}
```

```
Enter the number of nodes to create the circular linked list: 4
Enter data for node 1: 5
Enter data for node 2: 5
Enter data for node 3: 8
Enter data for node 4: 8
Original Circular Linked List: 5 5 8 8
Circular Linked List after removing duplicates: 5 8
```