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

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

```
Enter the number of nodes to create t
Enter data for node 1: 4
Enter data for node 2: 43
Enter data for node 3: 5
Enter data for node 4: 3
1. Insert at Beginning
2. Insert at End
3. Insert at Position
Delete at Beginning
5. Delete at End
6. Delete at Position
7. Search an Element
8. Display
9. Exit
Enter your choice:
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;
      cout << temp->data << " ";
      temp = temp->next;
    } while (temp != head);
    cout << endl;}
};
```

```
int main() {
                                                                 int data;
                                                                 cout << "Enter data for node " << i + 1 << ": ";
  CircularLinkedList cll;
                                                                 cin >> data;
  int n;
  cout << "Enter the number of nodes to create the
                                                                 insertAtEnd(data);
circular linked list: ";
                                                               }}
  cin >> n;
                                                             void insertAtEnd(int data) {
  cll.createList(n);
                                                               Node* newNode = new Node{data, nullptr};
  cout << "Original Circular Linked List: ";
                                                               if (!head) {
  cll.display();
                                                                 head = newNode;
  cll.reverse();
                                                                 newNode->next = head;
  cout << "Reversed Circular Linked List: ";
                                                               } else {
  cll.display();
                                                                 Node* temp = head;
  return 0;}
                                                                 while (temp->next != head) {
Enter the number of nodes to create
                                                                   temp = temp->next;
                                                                 }
Enter data for node 1: 4
                                                                 temp->next = newNode;
Enter data for node 2: 5
Enter data for node 3: 6
                                                                 newNode->next = head;
Original Circular Linked List: 4 5
                                                               }}
Reversed Circular Linked List: 6 5
                                                             int countNodes() {
3. Write a program to count the number of elements
                                                               if (!head) return 0;
of Circular Linked List
                                                               int count = 0;
#include <iostream>
                                                               Node* temp = head;
using namespace std;
                                                               do {
struct Node {
                                                                 count++;
  int data;
                                                                 temp = temp->next;
  Node* next;
                                                               } while (temp != head);
};
                                                               return count;}
class CircularLinkedList {
                                                             void display() {
  Node* head;
                                                               if (!head) return;
public:
                                                               Node* temp = head;
  CircularLinkedList() {
                                                               do {
    head = nullptr;}
                                                                 cout << temp->data << " ";
  void createList(int n) {
                                                                 temp = temp->next;
    if (n <= 0) return;
                                                               } while (temp != head);
    for (int i = 0; i < n; i++) {
```

```
cout << endl;}
                                                              }
};
                                                              void createList(int n) {
int main() {
                                                                if (n <= 0) return;
  CircularLinkedList cll;
                                                                for (int i = 0; i < n; i++) {
                                                                  float data;
  int n;
  cout << "Enter the number of nodes to create the
                                                                  cout << "Enter data for node " << i + 1 << ": ";
circular linked list: ";
                                                                  cin >> data;
  cin >> n;
                                                                  insertAtEnd(data);
  cll.createList(n);
                                                                }}
  cout << "Circular Linked List: ";
                                                              void insertAtEnd(float data) {
  cll.display();
                                                                Node* newNode = new Node{data, nullptr};
  int count = cll.countNodes();
                                                                if (!head) {
  cout << "Number of elements in the Circular Linked
                                                                  head = newNode;
List: " << count << endl;
                                                                  newNode->next = head;
  return 0;
                                                                } else {
}
                                                                  Node* temp = head;
 Enter the number of nodes to create
                                                                  while (temp->next != head) {
 Enter data for node 1: 4
 Enter data for node 2: 5
                                                                     temp = temp->next;
 Enter data for node 3: 8
                                                                  }
 Circular Linked List: 4 5 8
                                                                  temp->next = newNode;
 Number of elements in the Circular Li
                                                                  newNode->next = head;
4). Write a program to sort the elements Circular
                                                                }}
linked list.
                                                              void sortList() {
                                                                if (!head | | head->next == head) return;
#include <iostream>
                                                                Node* current = head;
using namespace std;
struct Node {
                                                                Node* index = nullptr;
                                                                int temp;
  float data;
                                                                do {
  Node* next;
                                                                  index = current->next;
};
                                                                  while (index != head) {
class CircularLinkedList {
                                                                     if (current->data > index->data) {
  Node* head;
                                                                       temp = current->data;
public:
                                                                       current->data = index->data;
  CircularLinkedList() {
                                                                       index->data = temp;
    head = nullptr;
```

```
}
         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:
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