NAME: MOAZZAM FAROOQUI

ROLLNO: CT-24068

COURSE CODE: CT-159

ASSIGNMENT: DSA LAB#02

INSTRUCTOR: SAYYDA SAHAR FATIMA

Q1.

SOURCE CODE:

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

class Node {
public:
    int val;
    Node*next;
    Node(int v){
        val=v;
        next=nullptr;
    }
};

Node* mergetwolists(Node*list1,Node*list2){
    if(list1==nullptr){
        return list2;
    }
    if(list2==nullptr){
        return list1;
    }

    if(list1->val<=list2->val){
        list1->next=mergetwolists(list1->next,list2);
        return list1;
    }
    else{
        list2->next=mergetwolists(list1,list2->next);
        return list2;
    }
}

void display(Node*head){
    while(head!=nullptr){
        cout<<head->val<<" ";
        head=head->next;
    }
    cout<<endl;
}
```

```
int main(void){
    Node*list1=new Node(1);
    list1->next=new Node(2);
    list1->next->next=new Node(4);

    Node*list2=new Node(1);
    list2->next=new Node(3);
    list2->next->next=new Node(4);

    Node*m=mergetwolists(list1,list2);
    cout<<"MERGED LIST:"<<endl;
    display(m);
    return 0;
}
```

OUTPUT:

```
MERGED LIST:
1 1 2 3 4 4

--------------------------------
Process exited after 0.3716 seconds with return value 0
Press any key to continue . . .
```

Q2.

SOURCE CODE:

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

class Node{
    public:
        Node*next;
        int val;

        Node(int v){
            next=nullptr;
            val=v;
        }
};
```

```
Node*deleteduplicates(Node*head){
    if(head==nullptr){
        return nullptr;
    }

    Node*current=head;
    while(current!=nullptr && current->next!=nullptr){
        if(current->val==current->next->val){
            Node*duplicate=current->next;
            current->next=current->next->next;
            delete duplicate;
        }
        else{
            current=current->next;
        }
    }
    return head;
}
```

```
void display(Node*head){
    while(head!=nullptr){
        cout<<head->val<<" ";
        head=head->next;
    }
    cout<<endl;
}

int main(void){
    Node*head=new Node(1);
    head->next=new Node(1);
    head->next->next=new Node(2);
    head->next->next->next=new Node(3);
    head->next->next->next->next=new Node(3);

    cout<<"ORIGINAL LIST:";
    display(head);

    head=deleteduplicates(head);

    cout<<"LIST AFTER REMOVING DUPLICATES:";
    display(head);

    return 0;
}
```

OUTPUT:

```
ORIGINAL LIST:1 1 2 3 3
LIST AFTER REMOVING DUPLICATES:1 2 3

--------------------------------
Process exited after 0.03034 seconds with return value 0
Press any key to continue . . .
```

Q3.

SOURCE CODE:

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

class Node{
    public:
        int val;
        Node*next;

        Node(int v){
            val=v;
            next=nullptr;
        }
};


Node*findmiddlenode(Node*head){
    if(head==nullptr || head->next==nullptr){
        return head;
    }
    Node*slow=head;
    Node*fast=head;

    while(fast->next!=nullptr && fast->next->next!=nullptr){
        slow=slow->next;
        fast=fast->next->next;
    }
    return slow;
}

Node*mergetwolists(Node*list1,Node*list2){
    if(list1==nullptr){
        return list2;
    }

    if(list2==nullptr){
        return list1;
    }

    if(list1->val<=list2->val){
        list1->next=mergetwolists(list1->next,list2);
        return list1;
    }
    else{
        list2->next=mergetwolists(list1,list2->next);
        return list2;
    }
}
```

```
Node*mergesort(Node*head){
    if(head==nullptr || head->next==nullptr){
        return head;
    }

    Node*middle=findmiddlenode(head);
    Node*righthalf=middle->next;
    middle->next=nullptr;

    Node*left=mergesort(head);
    Node*right=mergesort(righthalf);

    return mergetwolists(left,right);
}

void display(Node*head){
    while(head!=nullptr){
        cout<<head->val<<" ";
        head=head->next;
    }
    cout<<endl;
}
```

```
int main(void){
    Node*head=new Node(4);
    head->next=new Node(2);
    head->next->next=new Node(1);
    head->next->next->next=new Node(3);

    cout<<"ORIGINAL LIST:";
    display(head);

    head=mergesort(head);

    cout<<"SORTED LIST:";
    display(head);
    return 0;
}
```

OUTPUT:

```
ORIGINAL LIST:4 2 1 3
SORTED LIST:1 2 3 4

--------------------------------
Process exited after 0.3295 seconds with return value 0
Press any key to continue . . .
```

Q4.

SOURCE CODE:

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

class Node{
    public:
        int val;
        Node*next;

        Node(int v){
            val=v;
            next=nullptr;
        }
};

Node*reverse(Node*head){
    Node*prev=nullptr;
    Node*curr=head;
    while(curr!=nullptr){
        Node*next=curr->next;
        curr->next=prev;
        prev=curr;
        curr=next;
    }
    return prev;
}
```

```cpp
bool is_palindrome(Node*head){
    if(head==nullptr || head->next==nullptr){
        return true;
    }
    Node*slow=head;
    Node*fast=head;

    while(fast->next!=nullptr && fast->next->next!=nullptr){
        slow=slow->next;
        fast=fast->next->next;
    }

    Node*secondhalf=reverse(slow->next);

    Node*firsthalf=head;
    Node*tempsecond=secondhalf;
    bool palindrome=true;

    while(tempsecond!=nullptr){
        if(firsthalf->val!=tempsecond->val){
            palindrome=false;
            break;
        }
```

```
50          firsthalf=firsthalf->next;
51          tempsecond=tempsecond->next;
52      }
53      slow->next=reverse(secondhalf);
54      return palindrome;
55  }
56
57  int main(void){
58      Node*head=new Node(1);
59      head->next=new Node(2);
60      head->next->next=new Node(2);
61      head->next->next->next=new Node(1);
62
63      if(is_palindrome(head)){
64          cout<<"TRUE!"<<endl;
65      }
66      else{
67          cout<<"FALSE!"<<endl;
68      }
69      return 0;
70  }
```

OUTPUT:

```
TRUE!

--------------------------------
Process exited after 0.4632 seconds with return value 0
Press any key to continue . . .
```

Q5.

SOURCE CODE:

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

class Node{
    public:
        int data;
        Node*next;

        Node(int val){
            data=val;
            next=nullptr;
        }
};

class CircularQueue{
    private:
        Node*front;
        Node*rear;
    public:
        CircularQueue(){
            front=nullptr;
            rear=nullptr;
        }
```

```
        void enqueue(int val){
            Node*newnode=new Node(val);
            if(front==nullptr){
                front=newnode;
                rear=newnode;
                rear->next=front;
                return;
            }
            rear->next=newnode;
            rear=newnode;
            rear->next=front;
        }

        void dequeue(){
            if(isEmpty()){
                cout<<"QUEUE IS EMPTY!"<<endl;
                return;
            }
            if(front==rear){
                delete front;
                front=nullptr;
                rear=nullptr;
                return;
            }
```

```
            Node*temp=front;
            front=front->next;
            rear->next=front;
            delete temp;
        }

        int peek(){
            if(isEmpty()){
                cout<<"QUEUE IS EMPTY!"<<endl;
                return -1;
            }
            return front->data;
        }

        bool isEmpty(){
            return front==nullptr;
        }
```

```
        void display(){
            if(isEmpty()){
                cout<<"QUEUE IS EMPTY!"<<endl;
                return;
            }
            Node*temp=front;
            cout<<"QUEUE ELEMENTS:"<<endl;
            while(temp->next!=front){
                cout<<temp->data<<" ";
                temp=temp->next;
            }
            cout<<temp->data<<endl;
        }
};
```

```
int main(void){
    CircularQueue cq;
    cq.enqueue(1);
    cq.enqueue(2);
    cout<<"FRONT ELEMENT:"<<cq.peek()<<endl;
    cq.dequeue();
    cq.display();

    cq.enqueue(3);
    cq.display();
    return 0;
}
```

OUTPUT:

```
FRONT ELEMENT:1
QUEUE ELEMENTS:
2
QUEUE ELEMENTS:
2 3

--------------------------------
Process exited after 0.04328 seconds with return value 0
Press any key to continue . . .
```

Q6.

SOURCE CODE:

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

class Node{
    public:
       int data;
       Node*next;

       Node(int val){
          data=val;
          next=nullptr;
       }
};

class Stack{
    private:
        Node*top;
    public:
        Stack(){
            top=nullptr;
        }
        void push(int val){
            Node*newnode=new Node(val);
            newnode->next=top;
            top=newnode;
        }
```

```
        void pop(){
            if(isEmpty()){
                cout<<"STACK EMPTY.CANNOT UNDERFLOW"<<endl;
                return;
            }
            Node*temp=top;
            top=top->next;
            delete temp;
        }

        int peek(){
            if(isEmpty()){
                cout<<"STACK IS EMPTY!"<<endl;
                return -1;
            }
            return top->data;
        }

        bool isEmpty(){
            return top==nullptr;
        }
```

```
50          void display(){
51              if(isEmpty()){
52                  cout<<"STACK IS EMPTY!"<<endl;
53                  return;
54              }
55              Node*temp=top;
56              cout<<"STACK(TOP TO BOTTOM):"<<endl;
57              while(temp!=nullptr){
58                  cout<<temp->data<<" ";
59                  temp=temp->next;
60              }
61              cout<<endl;
62          }
63  };
```

```
65  int main(void){
66      Stack s;
67      s.push(1);
68      s.push(2);
69
70      cout<<"TOP ELEMENT:"<<s.peek()<<endl;
71
72      s.pop();
73      s.push(3);
74      s.display();
75      return 0;
76  }
```

OUTPUT:

```
TOP ELEMENT:2
STACK(TOP TO BOTTOM):
3 1

--------------------------------
Process exited after 0.4011 seconds with return value 0
Press any key to continue . . .
```