

NAME: MOAZZAM FAROOQUI

ROLLNO: CT-24068

COURSE CODE: CT-159

ASSIGNMENT: DSA LAB#02

INSTRUCTOR: SAYYDA SAHAR FATIMA

Q1.

SOURCE CODE:

```
1  #include<iostream>
2  using namespace std;
3
4  class Node {
5  public:
6      int val;
7      Node*next;
8      Node(int v){
9          val=v;
10         next=nullptr;
11     }
12 };
13
14 Node* mergetwolists(Node*list1,Node*list2){
15     if(list1==nullptr){
16         return list2;
17     }
18     if(list2==nullptr){
19         return list1;
20     }
21
22     if(list1->val<=list2->val){
23         list1->next=mergetwolists(list1->next,list2);
24         return list1;
25     }
26
27     else{
28         list2->next=mergetwolists(list1,list2->next);
29         return list2;
30     }
31 }
32
33 void display(Node*head){
34     while(head!=nullptr){
35         cout<<head->val<<" ";
36         head=head->next;
37     }
38     cout<<endl;
39 }
```

```

41 int main(void){
42     Node*list1=new Node(1);
43     list1->next=new Node(2);
44     list1->next->next=new Node(4);
45
46     Node*list2=new Node(1);
47     list2->next=new Node(3);
48     list2->next->next=new Node(4);
49
50     Node*m=mergetwolists(list1,list2);
51     cout<<"MERGED LIST:"<<endl;
52     display(m);
53     return 0;
54 }

```

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:

```

1  #include<iostream>
2  using namespace std;
3
4  class Node{
5      public:
6          Node*next;
7          int val;
8
9      Node(int v){
10         next=nullptr;
11         val=v;
12     }
13 };

```

```

15 Node*deleteduplicates(Node*head){
16     if(head==nullptr){
17         return nullptr;
18     }
19
20     Node*current=head;
21     while(current!=nullptr && current->next!=nullptr){
22         if(current->val==current->next->val){
23             Node*duplicate=current->next;
24             current->next=current->next->next;
25             delete duplicate;
26         }
27         else{
28             current=current->next;
29         }
30     }
31     return head;
32 }

```

```

34 void display(Node*head){
35     while(head!=nullptr){
36         cout<<head->val<<" ";
37         head=head->next;
38     }
39     cout<<endl;
40 }
41
42 int main(void){
43     Node*head=new Node(1);
44     head->next=new Node(1);
45     head->next->next=new Node(2);
46     head->next->next->next=new Node(3);
47     head->next->next->next->next=new Node(3);
48
49     cout<<"ORIGINAL LIST:";
50     display(head);
51
52     head=deleteduplicates(head);
53
54     cout<<"LIST AFTER REMOVING DUPLICATES:";
55     display(head);
56
57     return 0;
58 }

```

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:

```
1  #include<iostream>
2  using namespace std;
3
4  class Node{
5      public:
6          int val;
7          Node*next;
8
9      Node(int v){
10         val=v;
11         next=nullptr;
12     }
13 };
14
15
16 Node*findmiddlenode(Node*head){
17     if(head==nullptr || head->next==nullptr){
18         return head;
19     }
20     Node*slow=head;
21     Node*fast=head;
22
23     while(fast->next!=nullptr && fast->next->next!=nullptr){
24         slow=slow->next;
25         fast=fast->next->next;
26     }
27
28     return slow;
29 }
30 Node*mergetwolists(Node*list1,Node*list2){
31     if(list1==nullptr){
32         return list2;
33     }
34
35     if(list2==nullptr){
36         return list1;
37     }
38
39     if(list1->val<=list2->val){
40         list1->next=mergetwolists(list1->next,list2);
41         return list1;
42     }
43     else{
44         list2->next=mergetwolists(list1,list2->next);
45         return list2;
46     }
47 }
```

```

49 Node*mergesort(Node*head){
50     if(head==nullptr || head->next==nullptr){
51         return head;
52     }
53
54     Node*middle=findmiddlenode(head);
55     Node*righthalf=middle->next;
56     middle->next=nullptr;
57
58     Node*left=mergesort(head);
59     Node*right=mergesort(righthalf);
60
61     return mergetwolists(left,right);
62 }
63
64 void display(Node*head){
65     while(head!=nullptr){
66         cout<<head->val<<" ";
67         head=head->next;
68     }
69     cout<<endl;
70 }

```

```

72 int main(void){
73     Node*head=new Node(4);
74     head->next=new Node(2);
75     head->next->next=new Node(1);
76     head->next->next->next=new Node(3);
77
78     cout<<"ORIGINAL LIST:";
79     display(head);
80
81     head=mergesort(head);
82
83     cout<<"SORTED LIST:";
84     display(head);
85     return 0;
86 }

```

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:

```
1  #include<iostream>
2  using namespace std;
3
4  class Node{
5  public:
6      int val;
7      Node*next;
8
9      Node(int v){
10         val=v;
11         next=nullptr;
12     }
13 };
14
15 Node*reverse(Node*head){
16     Node*prev=nullptr;
17     Node*curr=head;
18     while(curr!=nullptr){
19         Node*next=curr->next;
20         curr->next=prev;
21         prev=curr;
22         curr=next;
23     }
24     return prev;
25 }
26
27 bool is_palindrome(Node*head){
28     if(head==nullptr || head->next==nullptr){
29         return true;
30     }
31     Node*slow=head;
32     Node*fast=head;
33
34     while(fast->next!=nullptr && fast->next->next!=nullptr){
35         slow=slow->next;
36         fast=fast->next->next;
37     }
38
39     Node*secondhalf=reverse(slow->next);
40
41     Node*firsthalf=head;
42     Node*tempsecond=secondhalf;
43     bool palindrome=true;
44
45     while(tempsecond!=nullptr){
46         if(firsthalf->val!=tempsecond->val){
47             palindrome=false;
48             break;
49         }
```



```

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:

```
1  #include<iostream>
2  using namespace std;
3
4  class Node{
5      public:
6          int data;
7          Node*next;
8
9      Node(int val){
10         data=val;
11         next=nullptr;
12     }
13 };
14
15 class CircularQueue{
16     private:
17         Node*front;
18         Node*rear;
19     public:
20     CircularQueue(){
21         front=nullptr;
22         rear=nullptr;
23     }
24
25     void enqueue(int val){
26         Node*newnode=new Node(val);
27         if(front==nullptr){
28             front=newnode;
29             rear=newnode;
30             rear->next=front;
31             return;
32         }
33         rear->next=newnode;
34         rear=newnode;
35         rear->next=front;
36     }
37
38     void dequeue(){
39         if(isEmpty()){
40             cout<<"QUEUE IS EMPTY!"<<endl;
41             return;
42         }
43         if(front==rear){
44             delete front;
45             front=nullptr;
46             rear=nullptr;
47             return;
48         }
```



```

49 |         Node*temp=front;
50 |         front=front->next;
51 |         rear->next=front;
52 |         delete temp;
53 |     }
54 |
55 |     int peek(){
56 |         if(isEmpty()){
57 |             cout<<"QUEUE IS EMPTY!"<<endl;
58 |             return -1;
59 |         }
60 |         return front->data;
61 |     }
62 |
63 |     bool isEmpty(){
64 |         return front==nullptr;
65 |     }

```

```

67 |     void display(){
68 |         if(isEmpty()){
69 |             cout<<"QUEUE IS EMPTY!"<<endl;
70 |             return;
71 |         }
72 |         Node*temp=front;
73 |         cout<<"QUEUE ELEMENTS:"<<endl;
74 |         while(temp->next!=front){
75 |             cout<<temp->data<<" ";
76 |             temp=temp->next;
77 |         }
78 |         cout<<temp->data<<endl;
79 |     }
80 | };

```

```

82 | int main(void){
83 |     CircularQueue cq;
84 |     cq.enqueue(1);
85 |     cq.enqueue(2);
86 |     cout<<"FRONT ELEMENT:"<<cq.peek()<<endl;
87 |     cq.dequeue();
88 |     cq.display();
89 |
90 |     cq.enqueue(3);
91 |     cq.display();
92 |     return 0;
93 | }

```

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:

```
1  #include<iostream>
2  using namespace std;
3
4  class Node{
5  public:
6      int data;
7      Node*next;
8
9      Node(int val){
10         data=val;
11         next=nullptr;
12     }
13 };
14
15 class Stack{
16 private:
17     Node*top;
18 public:
19     Stack(){
20         top=nullptr;
21     }
22     void push(int val){
23         Node*newnode=new Node(val);
24         newnode->next=top;
25         top=newnode;
26
27
28     void pop(){
29         if(isEmpty()){
30             cout<<"STACK EMPTY.CANNOT UNDERFLOW"<<endl;
31             return;
32         }
33         Node*temp=top;
34         top=top->next;
35         delete temp;
36     }
37
38     int peek(){
39         if(isEmpty()){
40             cout<<"STACK IS EMPTY!"<<endl;
41             return -1;
42         }
43         return top->data;
44     }
45
46     bool isEmpty(){
47         return top==nullptr;
48     }
```

```

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 . . .

```