

**NAME: MOAZZAM FAROOQUI**

**TASK: DSA ASSIGNMENT#01**

**INSTRUCTOR: SAMIA MASOOD AWAN**

**COURSE CODE: CT-159**

**ROLL NO: CT-24068**

**Question 1:**

Given an array of integers and a target sum, find two numbers in the array that add up to the target sum. Return their indices. You can assume there will be exactly one so

**SOURCE CODE:**

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  vector<int>sum_of_numbers(vector<int>&nums,int target){
5      int n=nums.size();
6      for(int i=0;i<n;i++){
7          for(int j=i+1;j<n;j++){
8              if(nums[i]+nums[j]==target){
9                  return {i,j};
10             }
11         }
12     }
13 }
14
15 int main(void){
16     vector<int>nums={10,20,30,40};
17     int target=40;
18
19     vector<int>m=sum_of_numbers(nums,target);
20     cout<<m[0]<<" "<<m[1]<<" "<<endl;
21     return 0;
22 }
```

**OUTPUT:**

0 2

-----  
Process exited after 0.6656 seconds with return value 0  
Press any key to continue . . .

### Question 2:

Create a function that takes a list of integers and returns True if the list contains duplicate elements, otherwise return False.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  bool containsDuplicate(vector<int>&nums){
5      if(nums.empty()) return false;
6
7      sort(nums.begin(),nums.end());
8      for(int i=0;i<nums.size()-1;i++){
9          if(nums[i]==nums[i+1]){
10             return true;
11         }
12     }
13     return false;
14 }
15
16 int main(void){
17     vector<int>nums={10,13,15,12,13};
18     if(containsDuplicate(nums)){
19         cout<<"TRUE"<<endl;
20     } else{
21         cout<<"FALSE"<<endl;
22     }
23     return 0;
24 }
```

### OUTPUT:

TRUE

-----  
Process exited after 0.3834 seconds with return value 0  
Press any key to continue . . .

### Question 3:

Write a program to find the middle node of a singly linked list.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 SinglyLinkedList{
16     private:
17         Node*head;
18     public:
19         SinglyLinkedList(){
20             head=nullptr;
21         }
```

```
23     void insert(int val){
24         Node*newnode=new Node(val);
25         if(head==nullptr){
26             head=newnode;
27             return;
28         }
29         Node*temp=head;
30         while(temp->next!=nullptr){
31             temp=temp->next;
32         }
33         temp->next=newnode;
34     }
35
36     Node*find_middle_node(){
37         Node*slow=head;
38         Node*fast=head;
39
40         while(fast!=nullptr && fast->next!=nullptr){
41             slow=slow->next;
42             fast=fast->next->next;
43         }
44         return slow;
45     }
```

```

47 |         void display(){
48 |             Node*temp=head;
49 |             while(temp!=nullptr){
50 |                 cout<<temp->data<<" ";
51 |                 temp=temp->next;
52 |             }
53 |             cout<<endl;
54 |         }
55 |     };
56 |
57 | int main(void){
58 |     SinglyLinkedList sll;
59 |     sll.insert(10);
60 |     sll.insert(20);
61 |     sll.insert(30);
62 |     sll.insert(40);
63 |     sll.insert(50);
64 |     cout<<"ORIGINAL LIST:";
65 |     sll.display();
66 |
67 |     cout<<"MIDDLE NODE IS:"<<sll.find_middle_node()->data<<" ";
68 |
69 |     return 0;
70 | }

```

## OUTPUT:

ORIGINAL LIST:10 20 30 40 50

MIDDLE NODE IS:30

-----

Process exited after 0.7749 seconds with return value 0

Press any key to continue . . .

#### Question 4:

Implement a function to reverse a singly linked list. Your solution should update the pointers of the nodes and not just print the elements in reverse.

#### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 SinglyLinkedList{
16     private:
17         Node*head;
18     public:
19         SinglyLinkedList(){
20             head=nullptr;
21         }
22
23         void insert(int val){
24             Node*newnode=new Node(val);
25             if(head==nullptr){
26                 head=newnode;
27                 return;
28             }
29             Node*temp=head;
30             while(temp->next!=nullptr){
31                 temp=temp->next;
32             }
33             temp->next=newnode;
34         }
35
36         Node*reverse(){
37             Node*curr=head;
38             Node*prev=nullptr;
39             while(curr!=nullptr){
40                 Node*next=curr->next;
41                 curr->next=prev;
42                 prev=curr;
43                 curr=next;
44             }
45             head=prev;
46         }
47     }
```

```

48 □ void display(){
49     Node*temp=head;
50 □ while(temp!=nullptr){
51     cout<<temp->data<<" ";
52     temp=temp->next;
53 }
54     cout<<endl;
55 }
56 };
57
58 □ int main(void){
59     SinglyLinkedList sll;
60     sll.insert(10);
61     sll.insert(20);
62     sll.insert(30);
63     sll.insert(40);
64     sll.insert(50);
65
66     cout<<"BEFORE REVERSING:";
67     sll.display();
68
69     sll.reverse();
70
71     cout<<"AFTER REVERSING:";
72     sll.display();
73 }

```

## **OUTPUT:**

```

BEFORE REVERSING:10 20 30 40 50
AFTER REVERSING:50 40 30 20 10

```

```

-----
Process exited after 0.7013 seconds with return value 0
Press any key to continue . . . █

```

### Question 5:

Given the heads of two sorted singly linked lists, merge the two lists into a single sorted list. The new list should be made by splicing together the nodes of the first two lists.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 SinglyLinkedList{
16     private:
17         Node*head;
18     public:
19         SinglyLinkedList(){
20             head=nullptr;
21         }
22
23     void insert(int val){
24         Node*newnode=new Node(val);
25         if(head==nullptr){
26             head=newnode;
27             return;
28         }
29         Node*temp=head;
30         while(temp->next!=nullptr){
31             temp=temp->next;
32         }
33         temp->next=newnode;
34     }
```



```

36 Node*mergetwolists(Node*list1,Node*list2){
37     if(list1==nullptr){
38         return list2;
39     }
40     if(list2==nullptr){
41         return list1;
42     }
43     if(list1->data<=list2->data){
44         list1->next=mergetwolists(list1->next,list2);
45         return list1;
46     }
47     else{
48         list2->next=mergetwolists(list1,list2->next);
49         return list2;
50     }
51 }
52
53 void display(Node*n){
54     while(n!=nullptr){
55         cout<<n->data<<" ";
56         n=n->next;
57     }
58     cout<<endl;
59 }

```

```

61 Node*getHead(){
62     return head;
63 }
64 };
65
66 int main(void){
67     SinglyLinkedList list1,list2,sll;
68
69     list1.insert(10);
70     list1.insert(30);
71     list1.insert(50);
72
73     list2.insert(20);
74     list2.insert(40);
75     list2.insert(60);
76
77     cout<<"LIST 1:";
78     list1.display(list1.getHead());
79
80     cout<<"LIST 2:";
81     list2.display(list2.getHead());
82
83     Node*mergedhead=sll.mergetwolists(list1.getHead(),list2.getHead());
84     cout<<"MERGED LIST:";
85     sll.display(mergedhead);
86     return 0;
87 }

```

## **OUTPUT:**

```

LIST 1:10 30 50
LIST 2:20 40 60
MERGED LIST:10 20 30 40 50 60

```

```

-----
Process exited after 0.6227 seconds with return value 0
Press any key to continue . . .

```



### Question 6:

Given the head of a linked list, remove the n nodes from the list and return its head.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 LinkedList{
16     private:
17         Node*head;
18     public:
19         LinkedList(){
20             head=nullptr;
21         }
22
23     void insert(int val){
24         Node*newnode=new Node(val);
25         if(head==nullptr){
26             head=newnode;
27             return;
28         }
29         Node*temp=head;
30         while(temp->next!=nullptr){
31             temp=temp->next;
32         }
33         temp->next=newnode;
34     }
35
36     Node*delete_n_nodes(int n){
37         while(head!=nullptr && n>0){
38             Node*temp=head;
39             head=head->next;
40             temp->next=nullptr;
41             delete temp;
42             n--;
43         }
44         return head;
45     }
```

```

47 |         void display(){
48 |             Node*temp=head;
49 |             while(temp!=nullptr){
50 |                 cout<<temp->data<<" ";
51 |                 temp=temp->next;
52 |             }
53 |             cout<<endl;
54 |         }
55 |     };
56 |
57 | int main(void){
58 |     LinkedList ll;
59 |     cout<<"BEFORE DELETING N NODES:"<<endl;
60 |     ll.insert(1);
61 |     ll.insert(2);
62 |     ll.insert(3);
63 |     ll.insert(4);
64 |     ll.display();
65 |
66 |     cout<<"AFTER DELETING N NODES:"<<endl;
67 |     ll.delete_n_nodes(2);
68 |     ll.display();
69 |     return 0;
70 | }

```

## OUTPUT:

BEFORE DELETING N NODES:

1 2 3 4

AFTER DELETING N NODES:

3 4

-----  
 Process exited after 0.5555 seconds with return value 0  
 Press any key to continue . . .

### Question 7:

Write a function to insert a new node at a specific position (e.g., at the beginning, at the end, or after a given node) in a circular singly linked list.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 CircularLinkedList{
16     private:
17         Node*head;
18         Node*tail;
19
20     public:
21         CircularLinkedList(){
22             head=nullptr;
23             tail=nullptr;
24         }
25
26     void insert_at_head(int val){
27         Node*newnode=new Node(val);
28         if(head==nullptr){
29             head=newnode;
30             tail=newnode;
31             tail->next=head;
32             return;
33         }
34         newnode->next=head;
35         head=newnode;
36         tail->next=head;
37     }
38
39     void insert_at_tail(int val){
40         Node*newnode=new Node(val);
41         if(head==nullptr){
42             head=newnode;
43             tail=newnode;
44             tail->next=head;
45             return;
46         }
47         tail->next=newnode;
48         tail=newnode;
49         tail->next=head;
50     }
```

```

52 void insert_after_node(Node*prevnode,int val){
53     if(prevnode==nullptr){
54         return;
55     }
56     Node*newnode=new Node(val);
57     newnode->next=prevnode->next;
58     prevnode->next=newnode;
59
60     if(prevnode==tail){
61         tail=newnode;
62     }
63 }

```

```

65 Node*search(int val){
66     if(head==nullptr){
67         return nullptr;
68     }
69     Node*temp=head;
70     while(temp->next!=head){
71         if(temp->data==val){
72             return temp;
73         }
74         temp=temp->next;
75     }
76
77     if(temp->data==val){
78         return temp;
79     }
80     return nullptr;
81 }

```

```

83 void display(){
84     if(head==nullptr){
85         cout<<"LIST IS EMPTY!"<<endl;
86         return;
87     }
88     Node*temp=head;
89     while(true){
90         cout<<temp->data<<" ";
91         temp=temp->next;
92         if(temp==head){
93             break;
94         }
95     }
96     cout<<"(back to head)"<<endl;
97 }
98 };

```

```

100  int main(void){
101      CircularLinkedList cll;
102
103      cll.insert_at_head(10);
104      cll.insert_at_head(20);
105      cll.insert_at_tail(5);
106      cll.insert_at_tail(1);
107      cll.display();
108
109      Node*found=cll.search(10);
110      cll.insert_after_node(found,15);
111      cll.display();
112
113      found=cll.search(1);
114      cll.insert_after_node(found,25);
115      cll.display();
116
117      return 0;
118  }

```

## **OUTPUT:**

```

20 10 5 1 (back to head)
20 10 15 5 1 (back to head)
20 10 15 5 1 25 (back to head)

-----
Process exited after 0.6705 seconds with return value 0
Press any key to continue . . . █

```

### Question 8:

Implement a function to delete a node from a doubly linked list, given a pointer to the node to be deleted.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  class Node{
5      public:
6          int data;
7          Node*next;
8          Node*prev;
9
10     Node(int val){
11         data=val;
12         next=nullptr;
13         prev=nullptr;
14     }
15 };
```

```
17 class DoublyLinkedList{
18     private:
19         Node*head;
20         Node*tail;
21     public:
22         DoublyLinkedList(){
23             head=nullptr;
24             tail=nullptr;
25         }
26
27         void insert_at_head(int val){
28             Node*newnode=new Node(val);
29             if(head==nullptr){
30                 head=newnode;
31                 tail=newnode;
32                 return;
33             }
34             newnode->next=head;
35             head->prev=newnode;
36             head=newnode;
37 }
```



```
39 □ void insert_at_tail(int val){
40 □     Node*newnode=new Node(val);
41 □     if(head==nullptr){
42 □         head=newnode;
43 □         tail=newnode;
44 □         return;
45 □     }
46 □     tail->next=newnode;
47 □     newnode->prev=tail;
48 □     tail=newnode;
49 □ }
```

```
51 □ void delete_at_head(){
52 □     if(head==nullptr){
53 □         cout<<"LIST IS EMPTY!"<<endl;
54 □         return;
55 □     }
56 □     if(head==tail){
57 □         delete head;
58 □         head=nullptr;
59 □         tail=nullptr;
60 □         return;
61 □     }
62 □     head=head->next;
63 □     delete head->prev;
64 □     head->prev=nullptr;
65 □ }
```

```
67 □ void delete_at_tail(){
68 □     if(head==nullptr){
69 □         cout<<"LIST IS EMPTY!"<<endl;
70 □         return;
71 □     }
72 □     if(head==tail){
73 □         delete head;
74 □         head=nullptr;
75 □         tail=nullptr;
76 □         return;
77 □     }
78 □     tail=tail->prev;
79 □     delete tail->next;
80 □     tail->next=nullptr;
81 □ }
```



```

83 void insert_after_node(Node*prevnode,int val){
84     if(prevnode==nullptr){
85         return;
86     }
87     Node*newnode=new Node(val);
88     newnode->next=prevnode->next;
89     newnode->prev=prevnode;
90     if(prevnode->next!=nullptr){
91         prevnode->next->prev=newnode;
92     }
93     else{
94         tail=newnode;
95     }
96     prevnode->next=newnode;
97 }

```

```

99 void delete_node(Node*delnode){
100     if(delnode==nullptr){
101         return;
102     }
103     if(delnode==head){
104         delete_at_head();
105         return;
106     }
107     if(delnode==tail){
108         delete_at_tail();
109         return;
110     }
111     delnode->prev->next=delnode->next;
112     delnode->next->prev=delnode->prev;
113
114     delete delnode;
115 }

```

```

117 Node*search(int val){
118     Node*temp=head;
119     while(temp!=nullptr){
120         if(temp->data==val){
121             return temp;
122         }
123         temp=temp->next;
124     }
125     return nullptr;
126 }
127
128 void display(){
129     Node*temp=head;
130     while(temp!=nullptr){
131         cout<<temp->data<<"<->";
132         temp=temp->next;
133     }
134     cout<<"NULL";
135 }
136 };

```

```

138 □ int main(void){
139     DoublyLinkedList dll;
140     dll.insert_at_head(10);
141     dll.insert_at_head(20);
142     dll.insert_at_tail(5);
143     dll.insert_at_tail(1);
144     dll.display();
145     cout<<endl;
146
147     Node*found=dll.search(20);
148     dll.insert_after_node(found, 15);
149     dll.display();
150     cout<<endl;
151
152     found=dll.search(10);
153     dll.delete_node(found);
154     dll.display();
155     cout<<endl;
156
157     dll.delete_at_head();
158     dll.delete_at_tail();
159     dll.display();
160     cout<<endl;
161     return 0;
162 }

```

## **OUTPUT:**

```

20<->10<->5<->1<->NULL
20<->15<->10<->5<->1<->NULL
20<->15<->5<->1<->NULL
15<->5<->NULL

-----
Process exited after 0.5906 seconds with return value 0
Press any key to continue . . .

```

### Question 9:

Write a function to reverse a doubly linked list. Your solution should correctly update the next and prev pointers for each node.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  class Node{
5      public:
6          int data;
7          Node*next;
8          Node*prev;
9
10     Node(int val){
11         data=val;
12         next=nullptr;
13         prev=nullptr;
14     }
15 };
16
17 class DoublyLinkedList{
18     private:
19         Node*head;
20     public:
21         DoublyLinkedList(){
22             head=nullptr;
23         }
24
25         void insert(int val){
26             Node*newnode=new Node(val);
27             if(head==nullptr){
28                 head=newnode;
29                 return;
30             }
31             Node*temp=head;
32             while(temp->next!=nullptr){
33                 temp=temp->next;
34             }
35             temp->next=newnode;
36             newnode->prev=temp;
37 }
```

```

39 void display(){
40     Node*temp=head;
41     while(temp!=nullptr){
42         cout<<temp->data<<" ";
43         temp=temp->next;
44     }
45     cout<<endl;
46 }
47
48 void reverse(){
49     Node*curr=head;
50     Node*temp=nullptr;
51     while(curr!=nullptr){
52         temp=curr->prev;
53         curr->prev=curr->next;
54         curr->next=temp;
55         curr=curr->prev;
56     }
57     if(temp!=nullptr){
58         head=temp->prev;
59     }
60 }
61 };

```

```

63 int main(void){
64     DoublyLinkedList dll;
65     dll.insert(10);
66     dll.insert(20);
67     dll.insert(30);
68     dll.insert(40);
69     cout<<"BEFORE REVERSING:";
70     dll.display();
71     dll.reverse();
72     cout<<"AFTER REVERSING:";
73     dll.display();
74     return 0;
75 }

```

## OUTPUT:

```

BEFORE REVERSING:10 20 30 40
AFTER REVERSING:40 30 20 10

```

```

-----
Process exited after 0.03565 seconds with return value 0
Press any key to continue . . .

```

### Question 10:

Given a circular singly linked list, rotate it by k positions. For example, with n=7 and k=3, the list 1, 2, 3, 4, 5, 6, 7 becomes 5, 6, 7, 1, 2, 3, 4.

### SOURCE CODE:

```
1  #include<bits/stdc++.h>
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 CircularLinkedList{
16     private:
17         Node*head;
18         int size;
19
20     public:
21     CircularLinkedList(){
22         head=nullptr;
23         size=0;
24     }
25
26     void insert(int val){
27         Node*newnode=new Node(val);
28         if(head==nullptr){
29             head=newnode;
30             newnode->next=head;
31         }
32         else{
33             Node*temp=head;
34             while(temp->next!=head){
35                 temp=temp->next;
36             }
37             temp->next=newnode;
38             newnode->next=head;
39         }
40         size++;
41     }
42
43     void rotate(int k){
44         if(head==nullptr || k==0) return;
45         k=k%size;
46         int steps=size-k;
47         for(int i=0;i<steps;i++){
48             head=head->next;
49         }
50     }
```

```

52 | void display(){
53 |     if(head==nullptr) return;
54 |     Node*temp=head;
55 |     for(int i=0;i<size;i++){
56 |         cout<<temp->data<<" ";
57 |         temp=temp->next;
58 |     }
59 |     cout<<endl;
60 | }
61 | };
62 |
63 | int main(void){
64 |     CircularLinkedList cll;
65 |     for(int i=1;i<=7;i++){
66 |         cll.insert(i);
67 |     }
68 |
69 |     cout<<"ORIGINAL LIST:";
70 |     cll.display();
71 |
72 |     cll.rotate(3);
73 |
74 |     cout<<"ROTATED LIST:";
75 |     cll.display();
76 |     return 0;
77 | }

```

## **OUTPUT:**

```

ORIGINAL LIST:1 2 3 4 5 6 7
ROTATED LIST:5 6 7 1 2 3 4

```

```

-----
Process exited after 0.6137 seconds with return value 0
Press any key to continue . . .

```