



FATIMA JINNAH WOMEN UNIVERSITY

LAB#13

OPEN-ENDED -LAB

SUBMITTED BY: Tanzeela Asghar

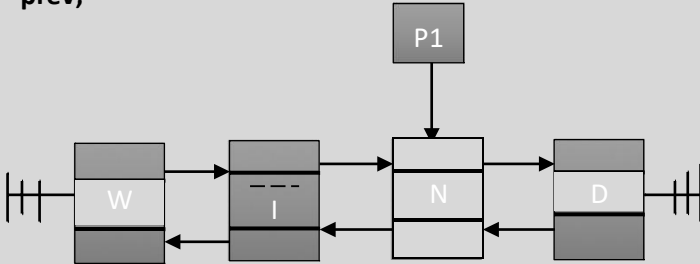
ROLL NO:2021-BSE-032

SUBMITTED TO: SIR REHAN AHMED SIDDIQUI

COURSE: DATA STRUCTURE AND ALGORITHM

TASK 1

Give answers to the following.

1.	The value of $10\ 2\ 5\ * +\ 3\ -$ is: <u>17</u>
2.	<p>What would be the contents of stack s after the following piece of code is executed:</p> <pre>Stack s; s.push(5); s.push(6); int i = s.top(); s.pop();</pre> <p>contents of stack: 5</p>
3.	<p>Assume the following doubly-linked list. Each node in the list is represented by the Node class listed as follows.</p> <pre>class Node{ public: char data; Node * next; Node * prev; };</pre>  <p>Using only the pointer p1 to access the list, write statements to:</p> <p>a. Display the contents of the four nodes in the list. (Hint: Write four statements).</p> <pre>p1->back->back->data; p1->back->data; p1->data; p1->next->data;</pre> <p>b. Replace 'W' by 'A' and 'D' by 'B'.</p> <pre>replace (char oldv, char newv) { p1=p1->back->back; //node p1 is defined while(p1-> >data!=oldv) { function inside the class</pre>

```

        p1=p1->next;
    }
    p1->data=newv;
} int
ma
{
    in()

    list l;                                //class object
    l.replace('W','A');
    l.replace('D','B');
    system("pause");
    return 0;
}

```

c. Insert a node containing value 'S' at the end of the list.

```

Node* ptr=new node;
Ptr->data='S';
P1->next->next=ptr;
Ptr->back=p1;
Ptr->next=NULL;

```

TASK 02

IMPLEMENT THE FOLLOWING FUNCTION DEFINITION ONLY.

EXERCISE 01

IMPLEMENT THE FUNCTION INT COUNTODD(NODE *HEAD) WHICH RETURNS THE TOTAL NUMBER OF NODES CONTAINING ODD VALUES IN A SINGLY LINKED LIST OF INTEGERS. THE ARGUMENT TO THE FUNCTION REPRESENTS A POINTER TO THE FIRST NODE OF THE LIST. THE NODE OF THE LINKED LIST IS IMPLEMENTED BY THE FOLLOWING CLASS.

```

class Node{ public: int data;
Node *next;
};
Class List{ Node
*head; public:
List()
{}
.....};

```

```

#include
"stdafx.h"
#include<iostream>
using namespace
std; class node {
public:
    int data;
    node *next;
}; class
List {
public:
    node *head;
List() {
head=NULL
; }
void insert(int x)
{
    node *temp=head;
    node *ptr=new node;
    ptr->data=x;
    ptr->next=NULL;
    if(head==NULL)
{
    head=ptr;
}
    else
{
    while(temp->next!
=NULL)
{
    temp=temp->next;
}
    temp->next=ptr;
}
}
void display()
{
    node *ptr=head;
    while(ptr!=NULL)
    {
        cout<<ptr->data<<" ";
        ptr=ptr->next;
    }
} void countodd(node*
head)
{
    int count=0;
node* ptr=head;
    while(ptr!=NULL)
    {
        if(ptr->data%2==1)
            count++;
        ptr=ptr->next;
    }
    cout<<"NUMBERS OF NODES HAVING ODD ELEMENTS IN IT ARE
"<<count<<endl; }
};
int _tmain(int argc, _TCHAR* argv[])
{

```

```

List l;
l.insert(1);
l.insert(2);
l.insert(3);
l.insert(4);
l.insert(5);
cout<<"NODES ELEMENTS
ARE"<<endl;    l.display();
cout<<endl;
    l.countodd(l.head);
cout<<endl;
system("pause");
    return 0;
}

```

```

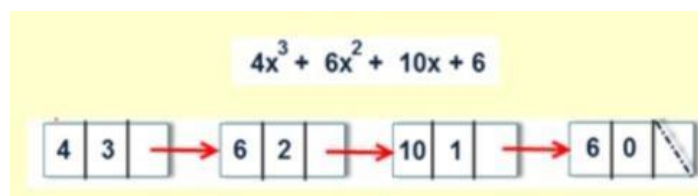
NODES ELEMENTS ARE
1 2 3 4 5
NUMBERS OF NODES HAVING ODD ELEMENTS IN IT ARE    3

Press any key to continue . . .

```

EXERCISE 02

A POLYNOMIAL CAN BE REPRESENTED AS A LINKED LIST. FOR EVERY TERM IN THE POLYNOMIAL THERE IS ONE ENTRY IN THE LINKED LIST CONSISTING OF THE TERM'S COEFFICIENT AND THE DEGREE AS ILLUSTRATED IN THE FOLLOWING.



THE FOLLOWING CLASS CAN BE EMPLOYED TO MODEL A NODE OF THE POLYNOMIAL LIST.

```

class PolyNode{ public: int coeff; int exponent; PolyNode * next;
};

```

COMPLETE THE FUNCTION INT EVALUATE(POLYNODE *HEAD, INT VAL) WHICH ACCEPTS A POINTER TO THE FIRST NODE OF A POLYNOMIAL LIST AND AN INTEGER VALUE. THE FUNCTION SHOULD EVALUATE THE POLYNOMIAL AT THE GIVEN VALUE. AS AN EXAMPLE, THE ABOVE POLYNOMIAL EVALUATED AT A T VALUE 2 SHOULD RETURN $4 * 2^3 + 6 * 2^2 + 10 * 2 + 6 * 2^0$. FOR SIMPLICITY, ASSUME THAT A FUNCTION POWER(INT X, INT N) IS AVAILABLE TO COMPUTE X^N

```

#include
"stdafx.h"
#include<iostream>
using namespace
std; class node {
public: double

```

```

co; double ex;
node * next; };
class List {
public:
    node *head;
    List()
    {
head=NULL;
    }
void insert_at_end(double c,double
e)
{
    node *temp=head;
node *ptr=new
node; ptr->co=c;
ptr->ex=e; ptr-
>next=NULL;
if(head==NULL)
{
    head=ptr;
}
else
{
    while (temp->next!=NULL)
    {
        temp=temp->next;
    }
    temp->next=ptr;
}
}

void display()
{
    node *temp=head;
cout<<" node contents
are : "<<endl; while (temp!=NULL)
{ cout<<"| "<<temp->co<<" "<<temp->ex<<" |"; temp=temp-
>next; } }
void eva(node *head,double
valueof_x)
{
    double sum=0.0;
node *temp;
temp=head;
while (temp!=NULL)
{
    sum=sum+temp->co*pow(valueof_x,temp-
>ex);
    temp=temp->next;
}
cout<<endl<<endl<<"sum of evaluated expression for "<<valueof_x<<"
is: "<<sum<<endl;
}
};
int _tmain(int argc, _TCHAR* argv[])
{ List
l;
l.insert_at_end(3.0,3.0);
l.insert_at_end(5.0,2.0);
l.insert_at_end(7.0,1.0);
l.insert_at_end(3.0,0.0);
l.display();
}

```

```

1.eva(1.head,2);
1.eva(1.head,3);
1.eva(1.head,4);
    system("pause");
    return 0;
}

```

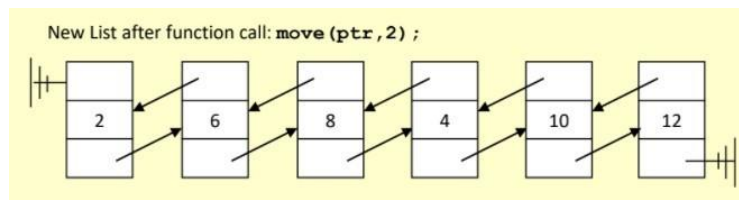
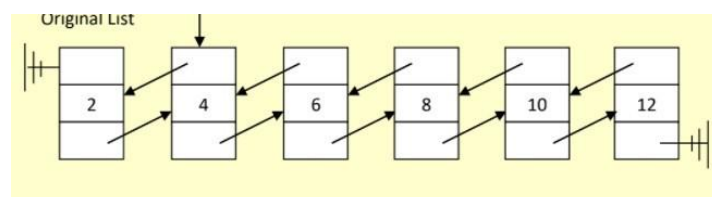
```

node contents are :
| 3    3    || 5    2    || 7    1    || 3    0    |
sum of evaluated expression for 2 is: 61
sum of evaluated expression for 3 is: 150
sum of evaluated expression for 4 is: 303
Press any key to continue . . .

```

EXERCISE 03

COMPLETE THE FUNCTION VOID MOVE(NODE *PTR, INT N) WHICH ACCEPTS A POINTER TO A NODE OF A DOUBLY LINKED LIST AND MOVES THE NODE 'N' POSITIONS FORWARD. FOR SIMPLICITY, ASSUME THAT THE VALUES OF 'PTR' AND 'N' ARE SUCH THAT THE FIRST AND LAST NODES DO NOT COME INTO PLAY. AN EXAMPLE ILLUSTRATION SHOWING THE ORIGINAL LIST AND THE LIST AFTER A SAMPLE FUNCTION CALL IS PRESENTED IN THE FOLLOWING. EACH NODE OF THE LIST HAS AN INTEGER (DATA) AND TWO POINTERS (NEXT AND PREV).



```

#include
"stdafx.h"
#include<iostream>
using namespace

```

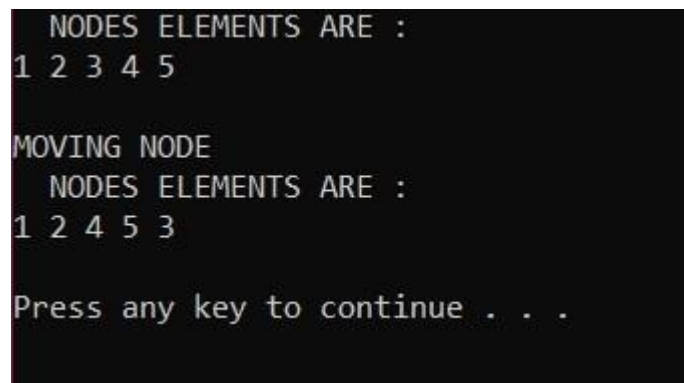
```

std; class node {
public:
    int data;
    node * next;
node
*prev; };
class List
{ public:
    node *head;
List() {
head=NULL
; }
void insert(int data)
{
    node *temp=head;    node
    *ptr=new node;      ptr-
    >data=data;          ptr-
    >next=NULL;          ptr-
    >prev=NULL;
    if(head==NULL)
    {
        head=ptr;
    }
    else
    {
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=ptr;
        ptr->prev=temp;
    } } void
display()
{
    node *temp=head;
cout<<"  NODES ELEMENTS
ARE : "<<endl; while(temp!=NULL)
{
cout<<temp->data<<" "; temp=temp-
>next;
}
cout<<endl<<endl;
}
void move(node *ptr,int n)
{
    ptr=head->next->next;
    node *temp=ptr;      ptr-
    >next->prev=ptr->prev;  ptr->prev-
    >next=ptr->next;
    for(int i=1;i<=n;i++)
    {
        temp=temp->next;
    } ptr->prev=temp;
    ptr->next=temp-
    >next; temp-
    >next=ptr;
} }
;
int _tmain(int argc, _TCHAR* argv[])
{ List
l;

```



```
l.insert(1);
l.insert(2);
l.insert(3);
l.insert(4);
l.insert(5);
l.display();
cout<<"MOVING
NODE"<<endl;
l.move(l.head,2);
l.display();
    system("pause");
    return 0;
}
```



```
    NODES ELEMENTS ARE :
1 2 3 4 5

MOVING NODE
    NODES ELEMENTS ARE :
1 2 4 5 3

Press any key to continue . . .
```
