



## **“Open ended lab”**

**COURSE :**

**DATA STURCTURES**

**SUBMITTED TO :**

**SIR RSHAN**

**SUBMITTED BY :**

**Rabia Batool (2021-BSE-064)**

**SECTION :**

**B**

## Data Structures and Algorithms

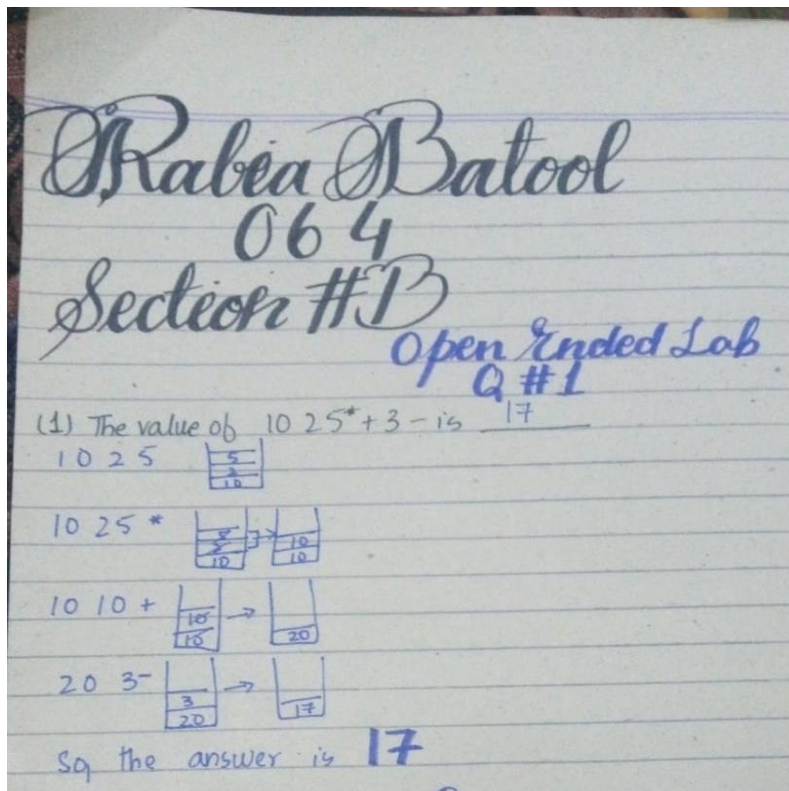
### Lab 11 OPEN ENDED LAB

**Note: Please note that understanding the question is the part of assignment so no queries**

Task 1 (Handwritten image):

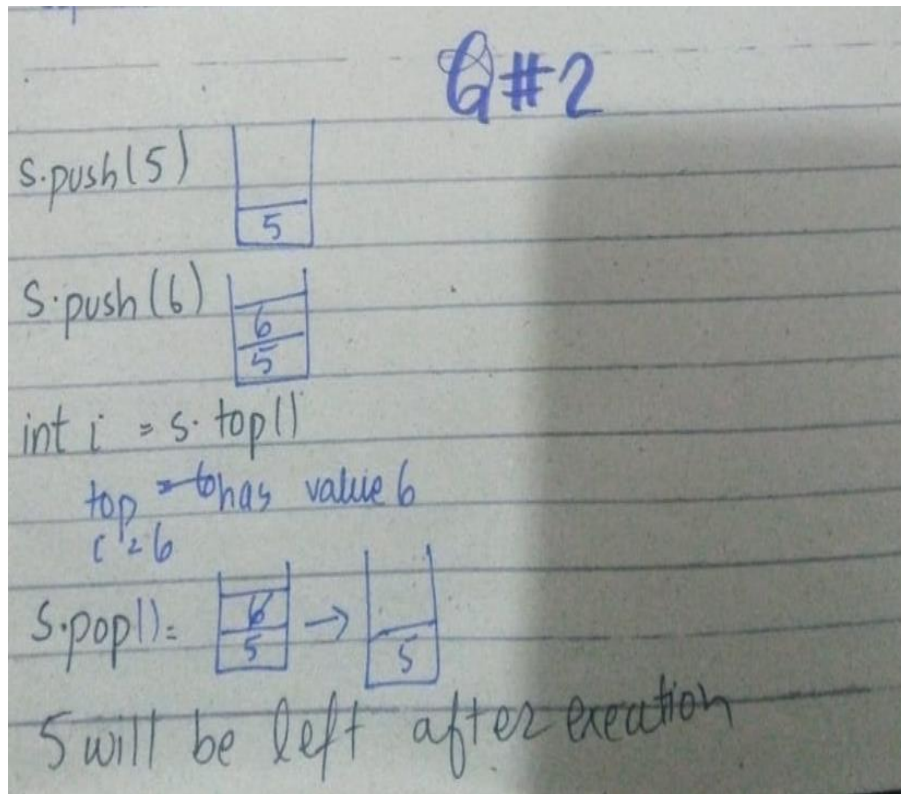
Give answers to the following.

1. The value of  $10 \ 2 \ 5^* + 3 -$  is: 17



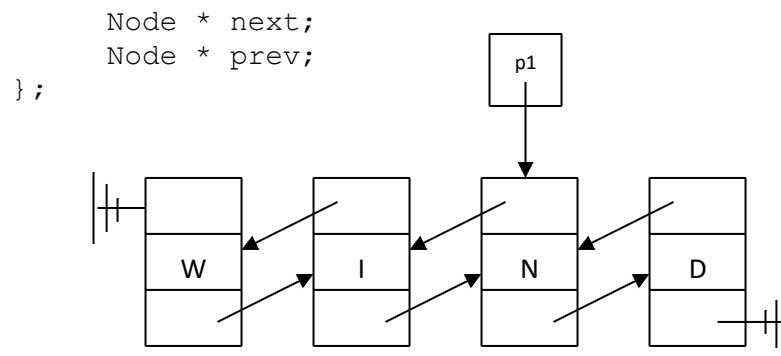
2. What would be the contents of stack s after the following piece of code is executed:

```
Stack s;  
s.push(5);  
s.push(6); int i  
= s.top();  
s.pop();
```



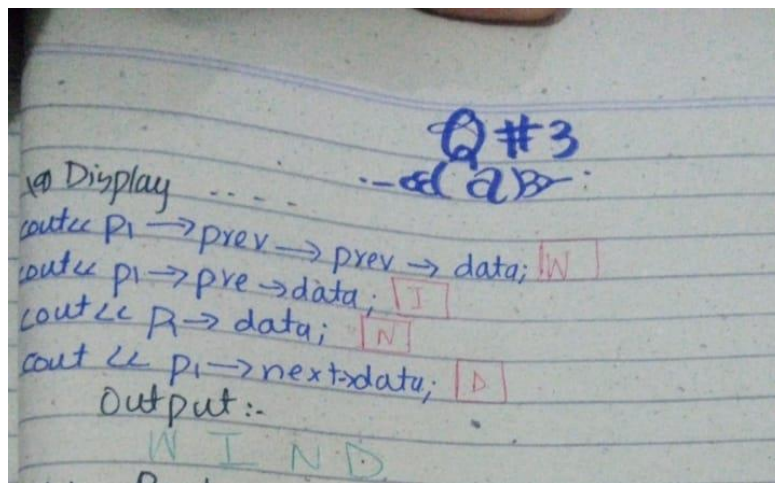
3. Assume the following doubly-linked list. Each node in the list is represented by the Node class listed as follows.

```
class Node{ public:  
    char data;
```



Using only the pointer p1 to access the list, write statements to:

- Display the contents of the four nodes in the list. (Hint: Write four statements).



b. Replace 'W' by 'A' and 'D' by 'B'.

(b) Replace - - -  
 $P_1 \rightarrow pre \rightarrow pre \rightarrow data = A;$   
 $P_1 \rightarrow next \rightarrow data = B;$

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

Insert:-  
 $Node *temp = new node;$   
 $temp \rightarrow data = S;$   
while  $P_1 \rightarrow next \rightarrow next = temp;$   
 $temp \rightarrow prev = P_1 \rightarrow next;$   
 $temp \rightarrow next = Null;$

## Task 2 :

Implement the following

# Exercise 1

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() {}
.....};
```

## Code:

```
// qaz.cpp : Defines the entry point for the console application.
//
```

```
#include "stdafx.h"
#include<iostream>
using namespace std;
```

```
class node
{public:
int data;
node *next;
```

```
};
class List
{public:
node *head;
List()
{head=NULL;
}
```

```

bool emptyList()
{if(head==NULL)
return true;
else
    return false;}

int CountOdd(node *head) {
    int count = 0;
    node *ptr= head;

    while (ptr != nullptr) {
        if (ptr->data % 2 != 0) {
            count++;
        }
        ptr = ptr->next;
    }

    return count;
}

void insert_begin(int value)
{node*ptr=new node;
ptr->data=value;
ptr->next=NULL;
ptr->next=head;
head=ptr;
}

void insert_end(int value)
{
    node* temp = new node;
    temp->data = value;
    temp->next = NULL;

    if (head == NULL) {
        head = temp;
    } else {
        node* s = head;
        while (s->next != NULL) {
            s = s->next;
        }
        s->next = temp;
    }
}

void traverse()
{node *ptr=head;
    while (ptr!=NULL)
    {cout<<ptr->data<<endl;
        ptr=ptr->next;
    }
}

};

```

```

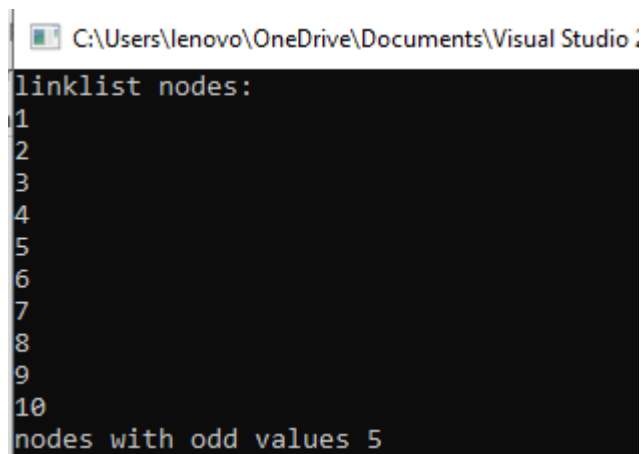
int _tmain(int argc, _TCHAR* argv[])
{
    List l;
    cout<<"linklist nodes:"<<endl;
    l.insert_begin(5);
    l.insert_begin(4);
    l.insert_begin(3);
    l.insert_begin(2);
    l.insert_begin(1);
    l.insert_end(6);
    l.insert_end(7);
    l.insert_end(8);
    l.insert_end(9);
    l.insert_end(10);
    l.traverse();
    int odd=l.CountOdd(l.head);
    cout<<"nodes with odd values "<<odd<<endl;

    system("pause");

    return 0;
}

```

## Output:



```

C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\linklist\linklist.cpp
linklist nodes:
1
2
3
4
5
6
7
8
9
10
nodes with odd values 5

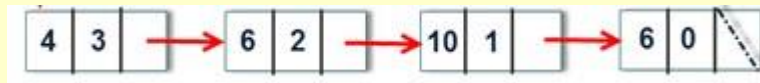
```

### Exercise 2



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.

$$4x^3 + 6x^2 + 10x + 6$$



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 value 2 should return  $4 * 2^3 + 6 * 2^2 + 10 * 2^1 + 6 * 2^0$ . For simplicity, assume that a function `power(int x, int n)` is available to compute  $x^n$ .

## Code:

```
// qaz.cpp : Defines the entry point for the console application.
//

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

class polynode
{public:
    int coeff;
    int exponent;
    polynode *next;
```

```

};
class List
{public:
polynode *head;
List()
{head=NULL;
}

bool emptyList()
{if(head==NULL)
return true;
else
return false;}
int power(int x, int n) {
if (n == 0) {
return 1;
} else {
return x * power(x, n - 1);
}
}

int Evaluate(polynode *head, int val) {
int result = 0;
polynode *temp = head;
int sol;
while (temp != nullptr) {
sol = temp->coeff * power(val, temp->exponent);
result=sol+result;
temp = temp->next;
}

return result;
}

void insert_begin(int a,int b)
{polynode*ptr=new polynode;
ptr->coeff=a;
ptr->exponent=b;
ptr->next=NULL;
ptr->next=head;
head=ptr;
}

void insert_end(int a,int b)
{
polynode* temp = new polynode;
temp->coeff = a;
temp->exponent = a;
temp->next = NULL;

if (head == NULL) {
head = temp;
} else {
polynode* s = head;
while (s->next != NULL) {
s = s->next;
}
}
}

```

```

        s->next = temp;
    }
}

void traverse()
{polynode *ptr=head;
cout<<"\npolynomial to evaluate:(";
    while (ptr!=NULL)
    {cout<<ptr->coeff<<"x^"<<ptr->exponent<<" + ";
        ptr=ptr->next;
    }cout<<")\n"<<endl;
}

};

int _tmain(int argc, _TCHAR* argv[])
{
    List l;
    l.insert_begin(0,6);
    l.insert_begin(10,1);
    l.insert_begin(6,2);
    l.insert_begin(4,3);

    l.traverse();
    int x = 2;
    int result =l. Evaluate(l.head, x);
    cout << "\n Polynomial evaluated at x = " << x << " is: " << result << endl<<endl;
    system("pause");
    return 0;

    system("pause");

    return 0;
}

```

## **Insertion of coefficient and exponent:**

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\qaz\Debug\qaz.exe
linklist nodes:
(coffecient,exponent):1,8
(coffecient,exponent):2,9
(coffecient,exponent):3,4
(coffecient,exponent):4,3
(coffecient,exponent):5,2
(coffecient,exponent):6,6
(coffecient,exponent):7,7
(coffecient,exponent):8,8
(coffecient,exponent):9,9
(coffecient,exponent):10,10
Press any key to continue . . .
```

## Result of polynomial:

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\qaz\Debug\qaz.exe
polynomial to evaluate:(4x^3 + 6x^2 + 10x^1 + 6x^0 )

Polynomial evaluated at x = 2 is: 82
Press any key to continue . . .
```

## Calculated result of polynomial:



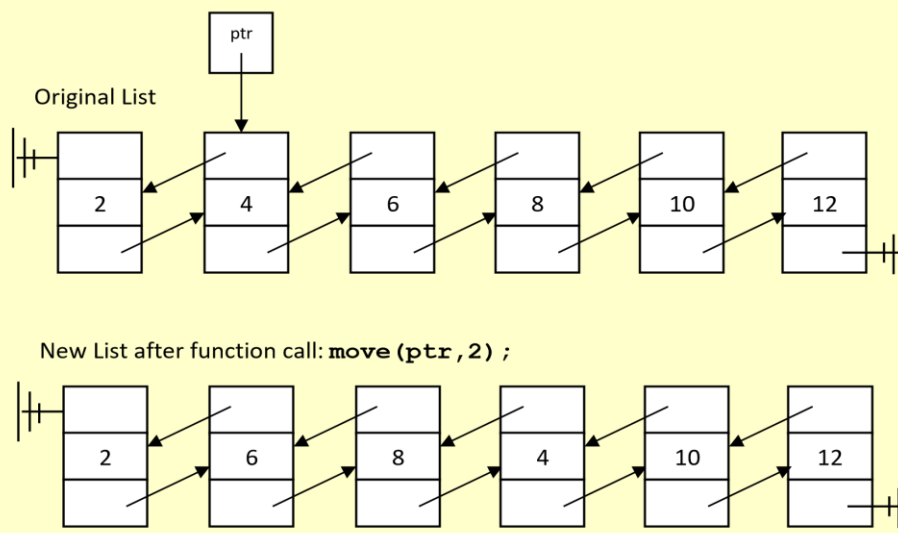
$$4(2)^3 + 6(2)^2 + 10(2)^1 + 6 =$$

82

|     |     |                |   |   |   |    |
|-----|-----|----------------|---|---|---|----|
| Rad | Deg | x!             | ( | ) | % | AC |
| Inv | sin | ln             | 7 | 8 | 9 | ÷  |
| π   | cos | log            | 4 | 5 | 6 | ×  |
| e   | tan | √              | 1 | 2 | 3 | −  |
| Ans | EXP | x <sup>y</sup> | 0 | . | = | +  |

## Exercise 3

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).



## Code:

```
// poiu.cpp : Defines the entry point for the console application.  
//
```

```
#include "stdafx.h"  
#include<iostream>  
using namespace std;  
  
class node  
{  
public:  
    int data;  
    node* next;  
    node* prev;  
};  
  
class DList  
{  
public:  
    node* head;  
  
    DList()  
    {  
        head = nullptr;  
    }  
  
    bool emptyList()  
    {  
        return head == nullptr;  
    }  
  
    void move(node* ptr, int n) {  
        if (ptr == nullptr || n == 0) {  
            cout << "Invalid parameters" << endl;  
            return;  
        }  
        else if(ptr->prev==NULL )  
            {cout<<"You can not move head \n"<<endl;}  
        else if(ptr->next==NULL )  
            {cout<<"You can not move last node\n";}   
        else if(ptr->next->next==NULL )  
            {cout<<"You can not move 2nd last node\n";}   
        else{node* s = head;  
  
            while (s != nullptr && s != ptr) {  
                s = s->next;  
            }  
  
            if (s == nullptr) {  
                cout << "Node not found in the list" << endl;  
                return;  
            }  
        }  
    }  
};
```

```

    }

    for (int i = 0; i < n; i++) {
        if (s != nullptr) {
            s = s->next;
        } else {
            cout << "Invalid position" << endl;
            return;
        }
    }

    ptr->prev->next = ptr->next;
    ptr->next->prev = ptr->prev;
    s->next->prev = ptr;
    ptr->next = s->next;
    s->next = ptr;
    ptr->prev = s;
}}

void insert_begin(int value)
{
    node* temp = new node;
    temp->data = value;

    if (head == nullptr)
    {
        head = temp;
        head->next = nullptr;
        head->prev = nullptr;
    }
    else
    {
        temp->next = head;
        if (head != nullptr)
            head->prev = temp;
        head = head->prev;
        head->prev = nullptr;
    }
}

void insert_end(int value)
{
    node* temp = new node;
    temp->data = value;

    if (head == nullptr)
    {
        head = temp;
        head->next = nullptr;
    }
    else
    {
        node* ptr = head;
        while (ptr->next != nullptr)
        {

```

```

        ptr = ptr->next;
    }
    ptr->next = temp;
    temp->prev = ptr;
    temp->next = nullptr;
}
}

void traverse()
{
    node* temp= head;
    while (temp!= nullptr)
    {
        cout << temp->data << endl;
        temp = temp->next;
    }
}

};

int _tmain(int argc, _TCHAR* argv[])
{DList d;
    cout << "insertions of nodes" << endl;
    d.insert_begin(12);
    d.insert_begin(10);
    d.insert_begin(8);
    d.insert_begin(6);
    d.insert_begin(4);
    d.insert_begin(2);
    d.traverse();

    node* ptr = d.head->next;
    int n = 2;
    d.move(ptr, n);
    cout << "Node 4 moved " << n << " positions forward " << endl;
    cout << "List after move movement: \n";
    d.traverse();
    system("pause");
    return 0;
}

```

## Movement of node:



```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\  
insertions of nodes  
2  
4  
6  
8  
10  
12  
Node 4 moved 2 positions forward  
List after move movement:  
2  
6  
8  
4  
10  
12  
Press any key to continue . . .
```

### **If user chose head:**

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Pr  
insertions of nodes  
2  
4  
6  
8  
10  
12  
You can not move head  
Press any key to continue . . .
```

### **If user chose last node:**

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\
insertions of nodes
2
4
6
8
10
12
You can not move last node
Press any key to continue . . .
```

**If user chose 2<sup>nd</sup> last node:**

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2
insertions of nodes
2
4
6
8
10
12
You can not move 2nd last node
Press any key to continue . . .
```

**Pointer is null:**

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\qa
insertions of nodes
2
4
6
8
10
12
Invalid parameters
Press any key to continue . . .
```

## Position is zero:

```
C:\Users\lenovo\OneDrive\Documents\Visual Studio 2010\Projects\qa
insertions of nodes
2
4
6
8
10
12
Invalid parameters
Press any key to continue . . .
```