Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

DATA STRUCTURE & ALGORITHM

LAB # 08

Submitted to:

Sir rehan

Submitted by:

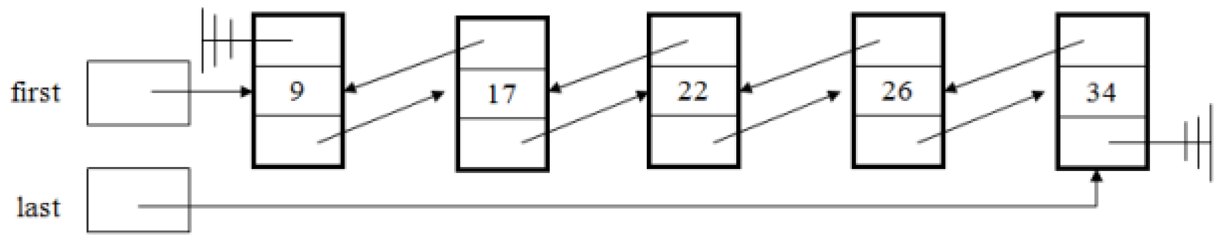
Rabia Batool

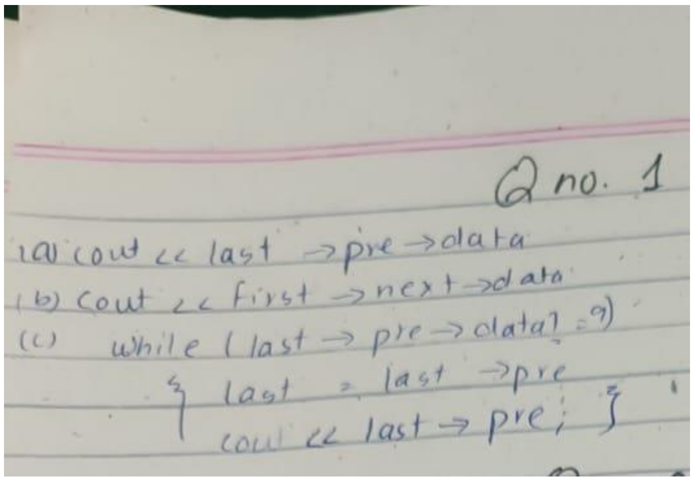
2022-BSE-064

Task 1 :

Give answers to the following.

Consider the following doubly linked list.





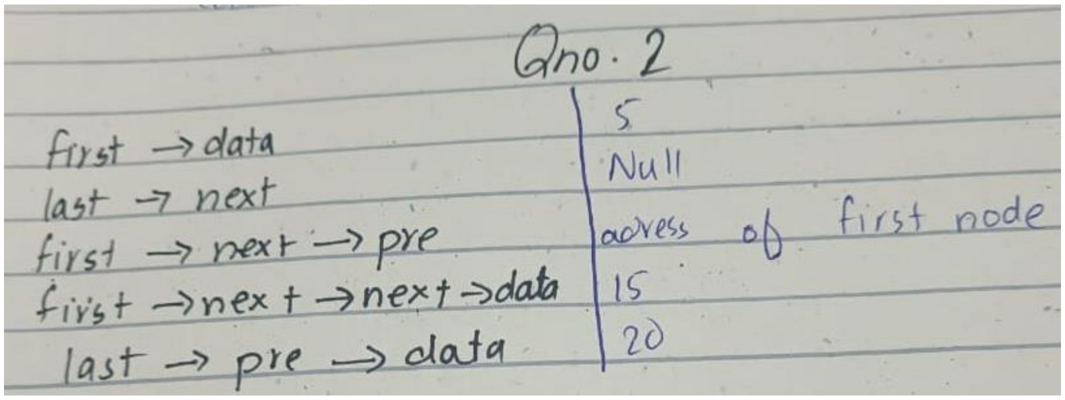
Q no. 1

```
(a) cout << last -> pre -> data
(b) cout << first -> next -> data
(c)    while (last -> pre -> data != 9)
          { last = last -> pre
            cout << last -> pre; }
```

Write C++ statements to:

a. Print the value 26 using the pointer 'last':

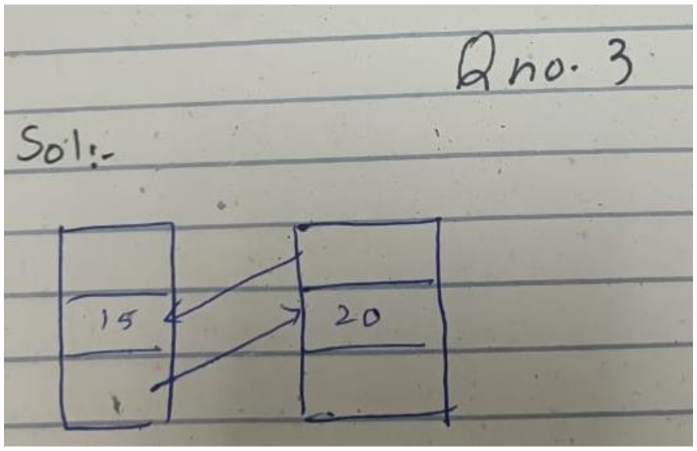
b. Print the value 17 using the pointer 'first:



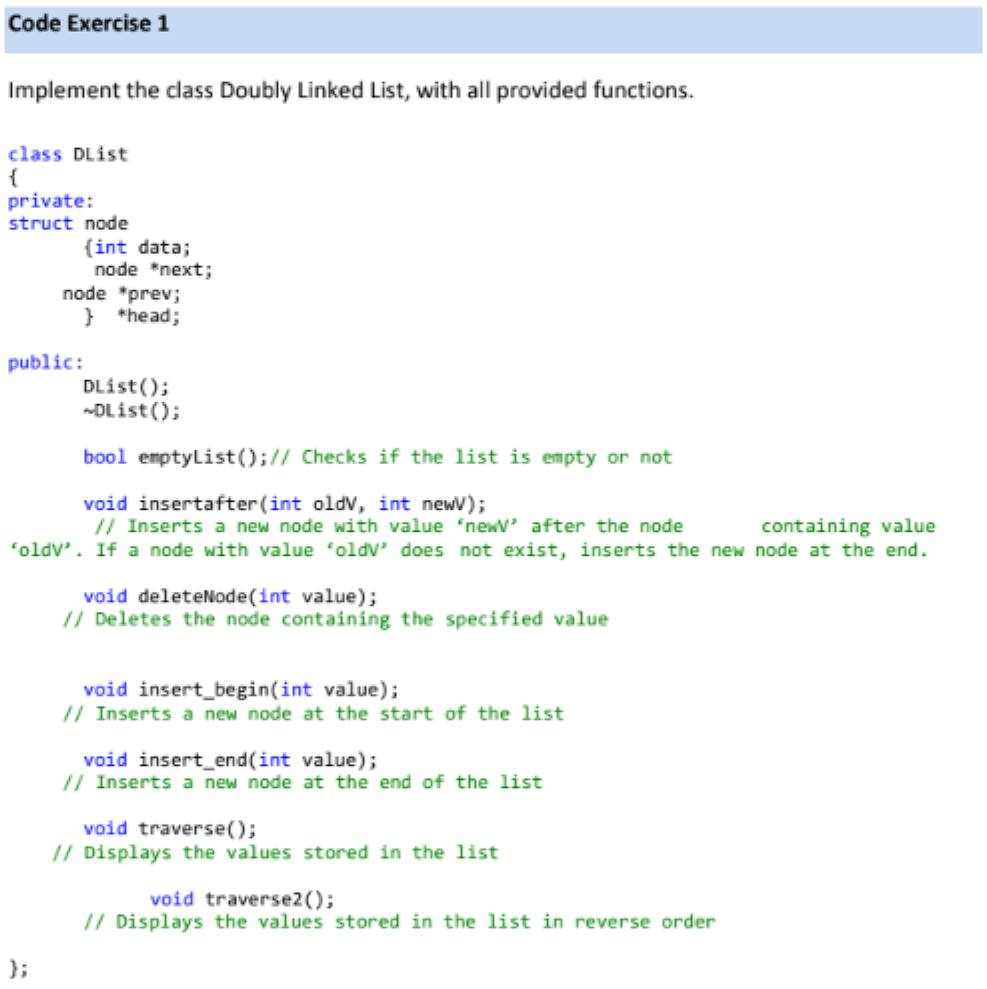
Q no. 2

first -> data	5
last -> next	Null
first -> next -> pre	adress of first node
first -> next -> next -> data	15
last -> pre -> data	20

c. Print the address of the node containing value 9 using the pointer 'last':



Task #1:



Code Exercise 1

Implement the class Doubly Linked List, with all provided functions.

```
class DList
{
private:
struct node
       {int data;
        node *next;
     node *prev;
       }  *head;

public:
       DList();
       ~DList();

       bool emptyList();// Checks if the list is empty or not

       void insertafter(int oldV, int newV);
        // Inserts a new node with value 'newV' after the node       containing value
'oldV'. If a node with value 'oldV' does not exist, inserts the new node at the end.

       void deleteNode(int value);
     // Deletes the node containing the specified value


       void insert_begin(int value);
     // Inserts a new node at the start of the list

       void insert_end(int value);
     // Inserts a new node at the end of the list

       void traverse();
    // Displays the values stored in the list

             void traverse2();
       // Displays the values stored in the list in reverse order

};
```

Code:

// Online C++ compiler to run C++ program online

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




class DList
{
private:
struct node
{int data;
node *next;
node *prev;
} *head;
public:
DList()
{head=NULL;}

bool emptyList()
{
   if (head == NULL)
      return true;
   else
      return false;
}
void insertafter(int oldV, int newV)
```

```
{
    node *s = head;
    int flag = 0;
    node *temp = new node;
    temp->data = newV;

    if (head == NULL)
    {
        head = temp;
    }
    else if (head->data == oldV && head->next == NULL)
    {
        head->next = temp;
        temp->next = NULL;
    }
    else
    {
        while (s != NULL && s->data != oldV)
        {
            s = s->next;
        }

        if (s != NULL)
        {
            temp->next = s->next;
            temp->prev = s;
            if (s->next != NULL)
            {
                s->next->prev = temp;
            }
            s->next = temp;
            flag++;
        }
    }

    if (flag == 0 && head != NULL)
    {
        s->next = temp;
        temp->prev = s;
        temp->next = NULL;
    }
}
void deleteNode(int value)
{
    int count = 0;
    node *s = head;

    if (head == NULL)
    {
        cout << "linklist is empty" << endl;
```

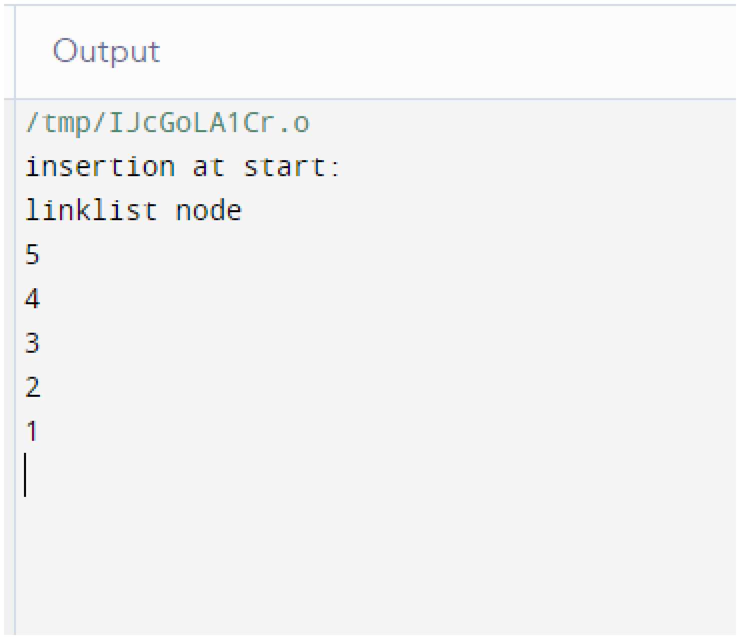
```
}
else if (head->data == value)
{
    node *temp = head;
    head = head->next;
    if (head != NULL)
    {
        head->prev = NULL;
    }
    temp->next = NULL;
    delete temp;
}
else
{
    while (s != NULL)
    {
        if (s->data == value)
        {
            s->prev->next = s->next;
            if (s->next != NULL)
            {
                s->next->prev = s->prev;
            }
            s->next = NULL;
            s->prev = NULL;
```

```
                delete s;
                count++;
                break;
            }
            else
            {
                s = s->next;
            }
        }
    }

    if (count == 0 && s != NULL)
    {
        s->prev = NULL;
        delete s;
    }
}
void insert_begin(int value)
{  node *temp = new node;
temp->data=value;
          if(head==NULL)
          { head=temp;
            head->next=NULL;
            head->prev=NULL;
          }
          else
          { temp->next=head;
            head->prev=temp;
            head=head->prev;
            head->prev=NULL;}
}
void insert_end(int value)
{ node *temp = new node;
temp->data=value;
          if(head==NULL)
          { head=temp;
            head->next=NULL;
          }
          else
          { node*ptr=head;
            while(ptr->next!=NULL)
            { ptr=ptr->next; }
            ptr->next=temp;
            temp->prev=ptr;
            temp->next=NULL;
          }
}
void traverse()
{
          node*s=head;
          while(s!=NULL)
          { cout<<s->data<<endl;
            s=s->next;
          }
}

void traverse2()
{
    if (head == NULL)
    {
        cout << "List is empty" << endl;
        return;
    }

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

    while (s != NULL)
    {
        cout << s->data << endl;
        s = s->prev;
    }
```

```
}

};




int main()
{DList d;
cout<<"insertions";
d.insert_begin(1);
d.insert_begin(2);
d.insert_begin(3);
d.insert_begin(4);
d.insert_end(6);
d.insert_end(12);
d.insert_end(10);

d.traverse();
cout<<"traverse2:"<<endl;
d.traverse2();
system("pause");

        return 0;
```

}

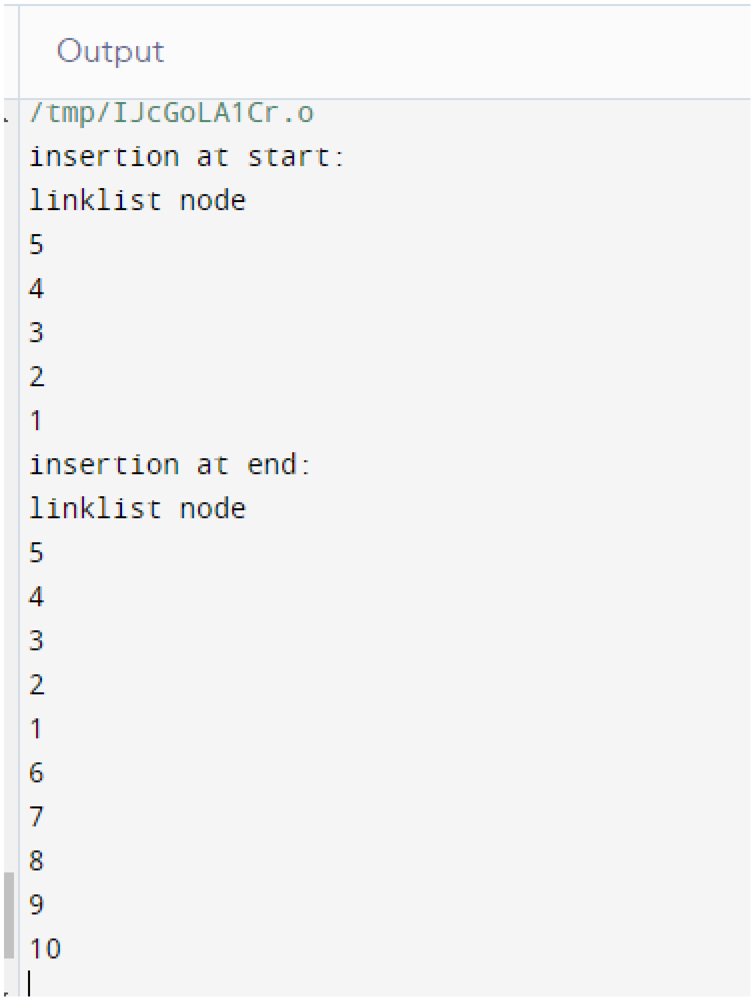
Insertion at start:



Output

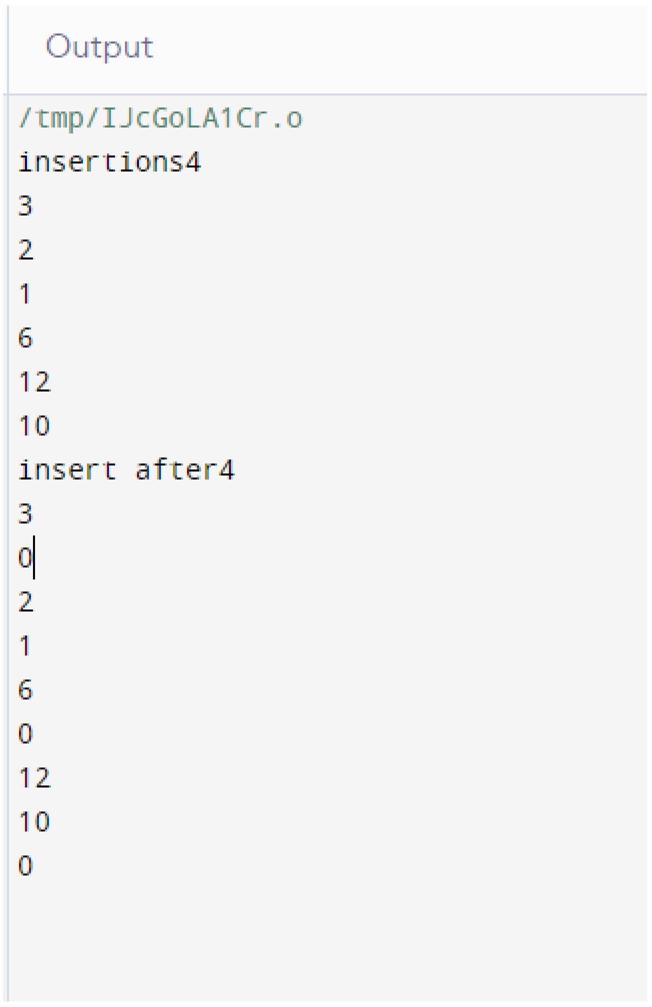
```
/tmp/IJcGoLA1Cr.o
insertion at start:
linklist node
5
4
3
2
1
```

Insertion at end:


Output

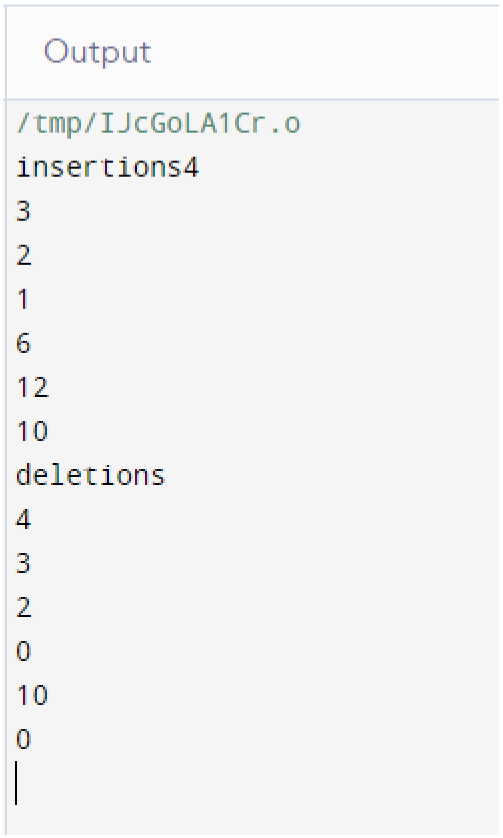
```
/tmp/IJcGoLA1Cr.o
insertion at start:
linklist node
5
4
3
2
1
insertion at end:
linklist node
5
4
3
2
1
6
7
8
9
10
```

<u>Insert after:</u>


Output

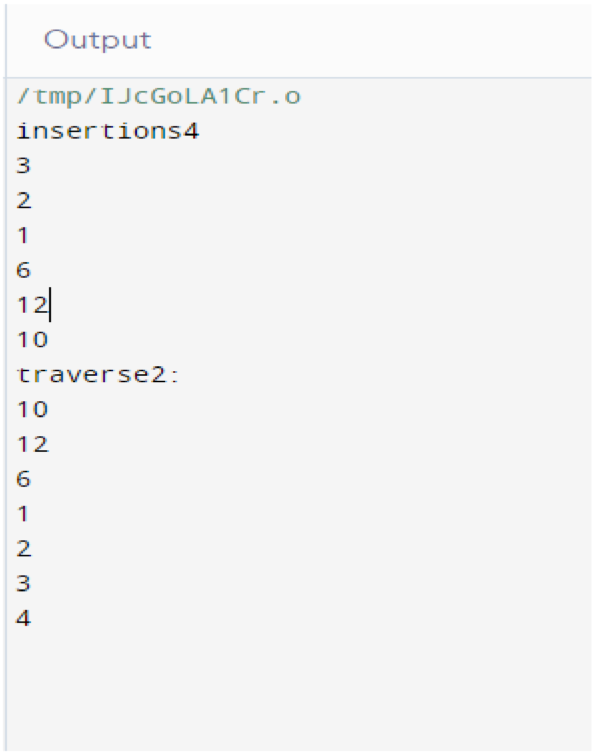
```
/tmp/IJcGoLA1Cr.o
insertions4
3
2
1
6
12
10
insert after4
3
0
2
1
6
0
12
10
0
```

<u>Deletions :</u>


Output

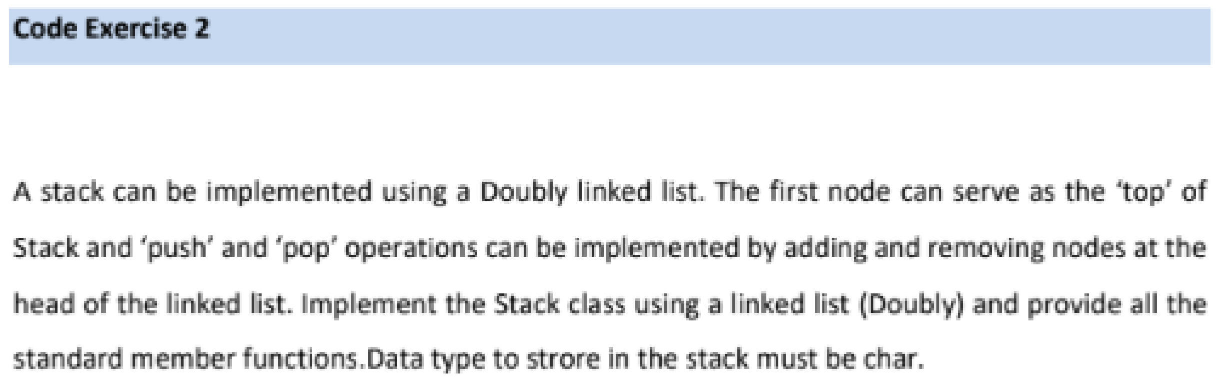
```
/tmp/IJcGoLA1Cr.o
insertions4
3
2
1
6
12
10
deletions
4
3
2
0
10
0
```

Traverse2:


Output

```
/tmp/IJcGoLA1Cr.o
insertions4
3
2
1
6
12
10
traverse2:
10
12
6
1
2
3
4
```

Task #2:



Code Exercise 2

A stack can be implemented using a Doubly linked list. The first node can serve as the 'top' of Stack and 'push' and 'pop' operations can be implemented by adding and removing nodes at the head of the linked list. Implement the Stack class using a linked list (Doubly) and provide all the standard member functions.Data type to strore in the stack must be char.

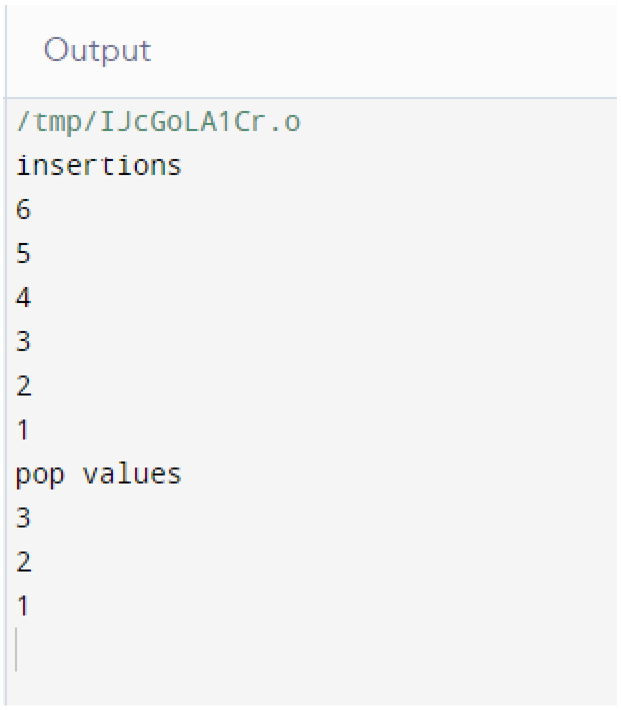
```
// Online C++ compiler to run C++ program online
#include<iostream>
using namespace std;



class DList
{
private:
struct node
{int data;
node *next;
node *prev;
} *top;
```

```
public:
DList()
{top=NULL;}

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

void pop()
{
    if (top == NULL)
    {
        cout << "linklist is empty" << endl;
    }
    else
    {
        node *temp = top;
        top = top->next;
        top->prev = NULL;
        temp->next = NULL;
        delete temp;
    }


}
void push(int value)
{  node *temp = new node;
temp->data=value;
        if(top==NULL)
        { top=temp;
          top->next=NULL;
          top->prev=NULL;
        }
        else
        { temp->next=top;
          top->prev=temp;
          top=top->prev;
          top->prev=NULL;}
}

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

```
}

};

int main()
{DList d;
cout<<"insertions"<<endl;
d.push(1);
d.push(2);
d.push(3);
d.push(4);
d.push(5);
d.push(6);
d.traverse();
cout<<"pop values"<<endl;
d.pop();
d.pop();
d.pop();
d.traverse();
system("pause");
```

```
        return 0;
}
```

Output:



```
Output
/tmp/IJcGoLA1Cr.o
insertions
6
5
4
3
2
1
pop values
3
2
1
```