LAB 04

M. Huzaifa Mustafa

Section AM

1. Simple LinkedList.
   Source code:

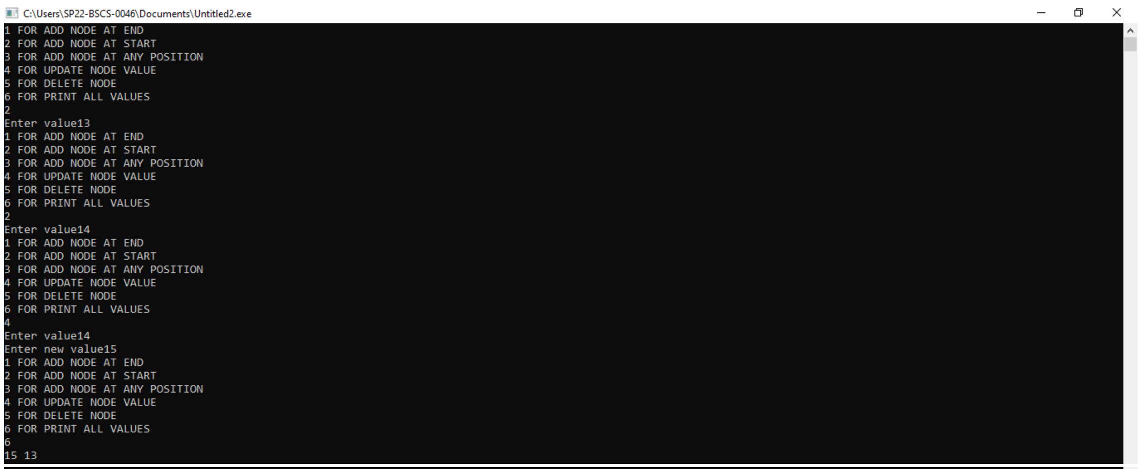
```
#include <iostream>
using namespace std;

class Node{
public:
int data;
Node*next;

Node(int data){
this->data = data;
this->next = NULL;
}
};

class LinkedList{
public:
Node*head;

LinkedList(){
this->head = NULL;
}

void addNode (int data){
Node*newNode = new Node(data);

if(head == NULL){
head= newNode;
}

else{
Node*current = head;

while(current->next!=NULL){
current = current->next;
}
current->next= newNode;
}
```

```
}

void addNodeAtStart(int data){
Node*newNode = new Node(data);
newNode->next = head;
head=newNode;
}

void insertNode(int data,int position){
Node*newNode = new Node(data);
Node*current = head;
int currentposition = 1;

while(current !=NULL && currentposition < position){
current = current ->next;
currentposition++;
}
if(current == NULL){
addNode(data);
}
else{
newNode->next = current->next;
current->next= newNode;
}
}

void updateNode(int oldVal, int newVal){
Node*current = head;
while(current!=NULL){
if(current ->data == oldVal){
current->data = newVal;
break;
}
current = current ->next;
}
}

void deleteNode(int val){
Node*current=head;
Node*previous= NULL;

while(current!=NULL && current ->data!=val){
previous = current;
current = current ->next;
```

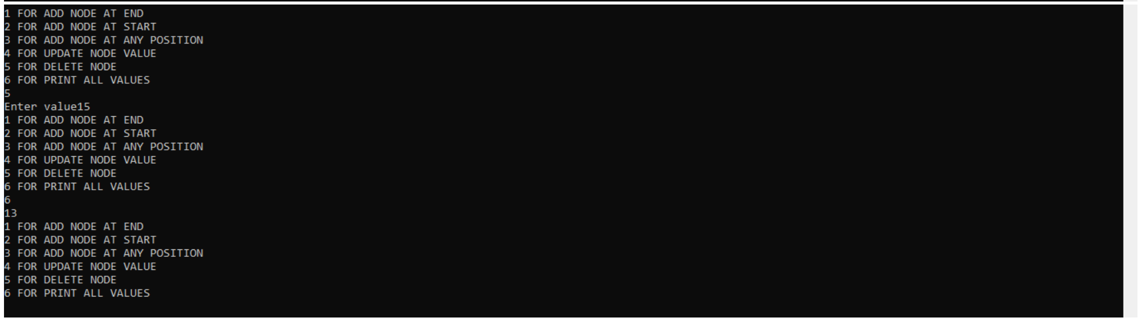
```
}
if(current!=NULL){
if(current ==head){
head = head->next;
}
else{
previous->next= current->next;

delete current;}
}
}

void printList(){
Node*current= head;
while (current!=NULL){
cout<<current->data<<" ";
current = current->next;
}
cout<<endl;
}
};

int main(){
LinkedList list;
int option =0;
int value=0;
int place=0;

do{
cout<<"1 FOR ADD NODE AT END\n";
cout<<"2 FOR ADD NODE AT START\n";
cout<<"3 FOR ADD NODE AT ANY POSITION\n";
cout<<"4 FOR UPDATE NODE VALUE\n";
cout<<"5 FOR DELETE NODE\n";
cout<<"6 FOR PRINT ALL VALUES\n";
cin>>option;

if(option ==1){
cout<<"Enter value";
cin>>value;
list.addNode(value);
}
else if(option ==2){
cout<<"Enter value";
```

```
cin>>value;
list.addNodeAtStart(value);
}
else if(option ==3){
cout<<"Enter value";
cin>>value;
cout<<"Enter position";
cin>>place;
list.insertNode(value,place);
}
else if(option ==4){
cout<<"Enter value";
cin>>value;
cout<<"Enter new value";
cin>>place;
list.updateNode(value,place);
}
else if(option==5){
cout<<"Enter value";
cin>>value;
list.deleteNode(value);
}
else if(option==6){
list.printList();
}
else{
cout<<"Incorrect option";
}
}
while(option!=0);
};
```

Picture:



```
C:\Users\SP22-BSCS-0046\Documents\Untitled2.exe
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
2
Enter value13
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
2
Enter value14
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
4
Enter value14
Enter new value15
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
6
15 13
```



```
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
5
Enter value15
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
6
13
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
```

2. Doubly LinkedList.
   Source code:

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

class Node{
public:
int data;
Node* next;
Node* previous;

Node(int data){
this->data=data;
this->next=NULL;
this->previous=NULL;
}

};

class LinkedList{
public:
Node* head;
```

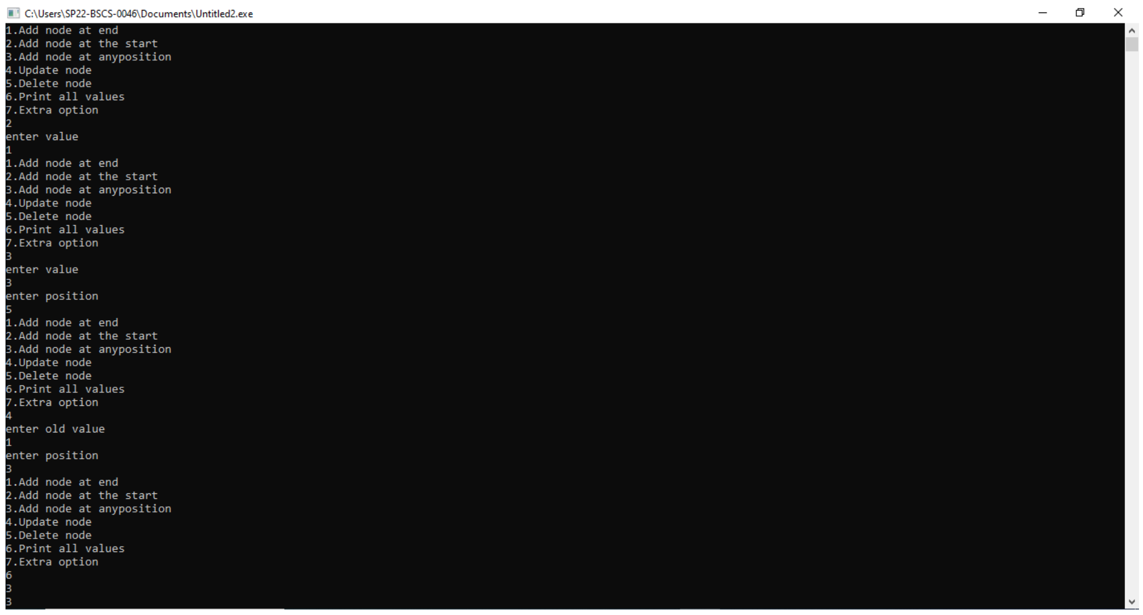
```
LinkedList(){
this->head=NULL;
}

void addNode(int data){
Node *n1=new Node(data);
if(head==NULL){
head=n1;
}
else
{
Node* current=head;
while(current->next!=NULL){
current=current->next;
n1->previous=current;
}
current->next=n1;
}
}

void addnodeatstart(int data){
Node *n1=new Node(data);
n1->next=head;
head=n1;

}

void insertNode(int data,int position){
Node* n1=new Node(data);
Node* current=head;
int currentposition=1;

while(current!=NULL && currentposition<position){
current=current->next;
currentposition++;
}

if(current==NULL){
addNode(data);
}
else
{
n1->next=current->next;
current->next=n1;
```

```
}
}

void updateNode(int oldval,int newval){
Node* current=head;

while(current!=NULL){
if(current->data=oldval){
current->data=newval;
break;
}
current=current->next;
}
}

void deleteNode(int val){
Node *current=head;
Node *previous=NULL;
while(current!=NULL && current->data!=val){
previous=current;
current=current->next;
}

if(current!=NULL){
if(current!=NULL){
if(current==head){
head=head->next;
}

else
{
previous->next=current->next;
}

delete current;
}
}
}

void PrintLinkedList(){
Node* current=head;
while(current!=NULL){
cout << current->data << endl;
current=current->next;
```

```
}
}

void printselected(int demo){
Node* current=head;
while(current!=NULL){
if(current->data==demo){
current=current->previous;
cout << current->data << endl;
current=current->next;
cout << current->data << endl;
current=current->next;
cout << current->data << endl;
}
current=current->next;
}
}

};



int main(){
LinkedList list;
int option=0;
int value=0;
int place=0;

do{
cout << "1.Add node at end\n2.Add node at the start\n3.Add node at anyposition\n4.Update
node\n5.Delete node\n6.Print all values\n7.Extra option" << endl;
cin>>option;
if(option==1){
cout << "enter value" << endl;
cin>>value;
list.addNode(value);
}

else if(option==2){
cout << "enter value" << endl;
cin>>value;
list.addnodeatstart(value);
}
```

```
else if(option==3){
cout << "enter value" << endl;
cin>>value;
cout << "enter position" << endl;
cin>>place;
list.insertNode(value,place);
}

else if(option==4){
cout << "enter old value" << endl;
cin>>value;
cout << "enter position" << endl;
cin>>place;
list.updateNode(value,place);
}

else if(option==5){
cout << "enter value" << endl;
cin>>value;
list.deleteNode(value);
}
else if(option==6){
list.PrintLinkedList();
}

else if(option==7){
cout << "enter value" << endl;
cin>>value;;
list.printselected(value);
}

}

while(option!=0);
return 0;
}
```

Picture:

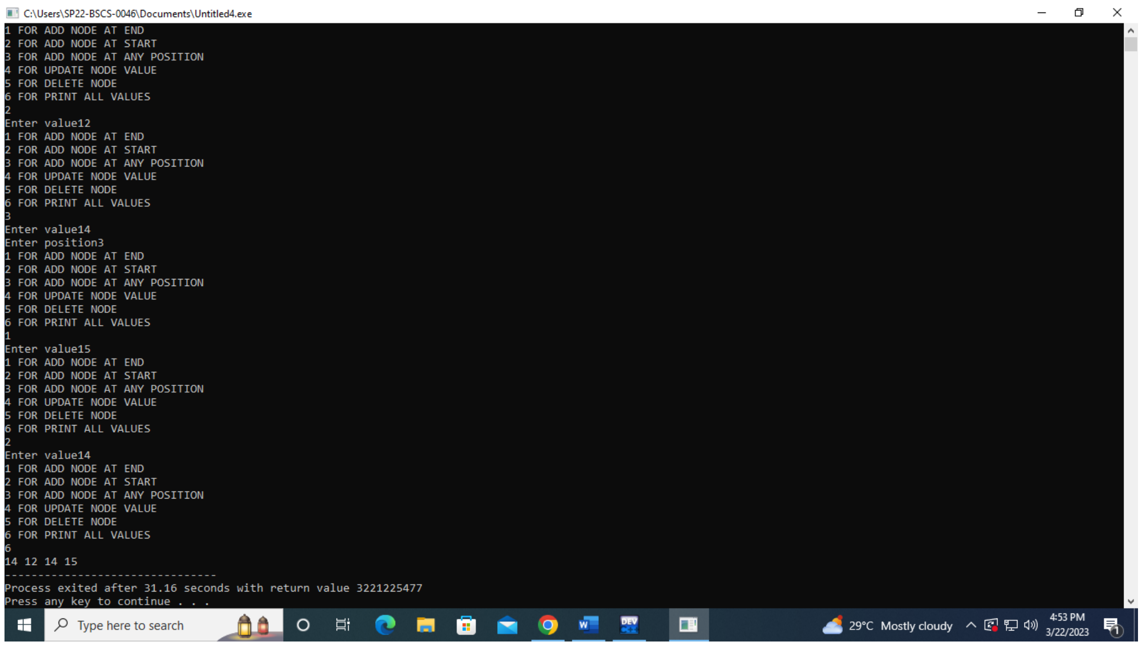

```
C:\Users\SP22-BSCS-0046\Documents\Untitled2.exe
1.Add node at end
2.Add node at the start
3.Add node at anyposition
4.Update node
5.Delete node
6.Print all values
7.Extra option
2
enter value
1
1.Add node at end
2.Add node at the start
3.Add node at anyposition
4.Update node
5.Delete node
6.Print all values
7.Extra option
3
enter value
3
enter position
5
1.Add node at end
2.Add node at the start
3.Add node at anyposition
4.Update node
5.Delete node
6.Print all values
7.Extra option
4
enter old value
1
enter position
3
1.Add node at end
2.Add node at the start
3.Add node at anyposition
4.Update node
5.Delete node
6.Print all values
7.Extra option
6
3
3
```

3. Circular LinkedList.
Source code:

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

class Node{
public:
int data;
Node*next;
Node*head;

Node(int data){
this->data = data;
this->next = head;
}
};

class LinkedList{
public:
Node*head;

LinkedList(){
this->head = NULL;
}

void addNode (int data){
Node*newNode = new Node(data);
```

```
if(head == NULL){
head = newNode;
}

else{
Node*current = head;

while(current->next!=NULL){
current = current->next;
}
current->next= newNode;
}
}

void addNodeAtStart(int data){
Node*newNode = new Node(data);
newNode->next = head;
head=newNode;
}

void insertNode(int data,int position){
Node*newNode = new Node(data);
Node*current = head;
int currentposition = 1;

while(current !=NULL && currentposition < position){
current = current ->next;
currentposition++;
}
if(current == NULL){
addNode(data);
}
else{
newNode->next = current->next;
current->next= newNode;
}
}

void updateNode(int oldVal, int newVal){
Node*current = head;
while(current!=NULL){
if(current ->data == oldVal){
current->data = newVal;
break;
}
current = current ->next;
}
}

void deleteNode(int val){
Node*current=head;
Node*previous= NULL;

while(current!=NULL && current ->data!=val){
previous = current;
current = current ->next;
}
if(current!=NULL){
if(current ==head){
head = head->next;
previous->next = head;
}
else{
previous->next= current->next;
delete current;}
}
}

void printList(){
Node*current= head;
while (current!=NULL){
cout<<current->data<<" ";
current = current->next;
}
cout<<endl;
}
};

int main(){
LinkedList list;
int option =0;
int value=0;
int place=0;

do{
cout<<"1 FOR ADD NODE AT END\n";
cout<<"2 FOR ADD NODE AT START\n";
cout<<"3 FOR ADD NODE AT ANY POSITION\n";
```

```
cout<<"4 FOR UPDATE NODE VALUE\n";
cout<<"5 FOR DELETE NODE\n";
cout<<"6 FOR PRINT ALL VALUES\n";
cin>>option;

if(option ==1){
cout<<"Enter value";
cin>>value;
list.addNode(value);
}
else if(option ==2){
cout<<"Enter value";
cin>>value;
list.addNodeAtStart(value);
}
else if(option ==3){
cout<<"Enter value";
cin>>value;
cout<<"Enter position";
cin>>place;
list.insertNode(value,place);
}
else if(option ==4){
cout<<"Enter value";
cin>>value;
cout<<"Enter new value";
cin>>place;
list.updateNode(value,place);
}
else if(option==5){
cout<<"Enter value";
cin>>value;
list.deleteNode(value);
}
else if(option==6){
list.printList();
}
else{
cout<<"Incorrect option";
}
}
while(option!=0);
};
```
Picture:



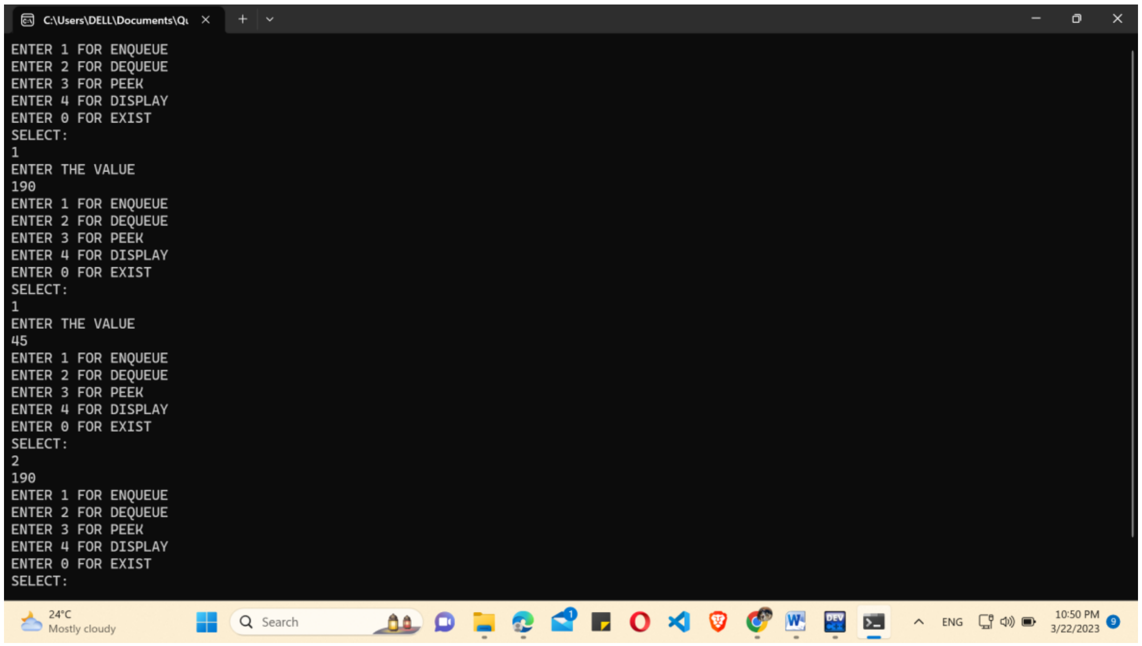
```
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
2
Enter value12
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
3
Enter value14
Enter position3
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
1
Enter value15
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
2
Enter value14
1 FOR ADD NODE AT END
2 FOR ADD NODE AT START
3 FOR ADD NODE AT ANY POSITION
4 FOR UPDATE NODE VALUE
5 FOR DELETE NODE
6 FOR PRINT ALL VALUES
6
14 12 14 15
--------------------------------
Process exited after 31.16 seconds with return value 3221225477
Press any key to continue . . .
```

4. Queue Using Linked list.
   Source code:

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

class Node {
	public:
		int data;
		Node* next;
	Node(int data){
		this->data = data;
		this->next = NULL;
	}

};


class Queue{
	public:
		Node* front;
		Node* rear;
	Queue(){
		front = NULL;
		rear = NULL;
	}

	void Enqueue(int value){
        Node* newNode = new Node(value);
        if(rear== NULL){
            front = rear = newNode;
        }
        else{
            rear->next = newNode;
            rear = newNode;
        }
    }

    int peek(){
        return front->data;
    }

    void Dequeue(){
        if(front == NULL){
            cout<<"Stack is Empty"<<endl;
        }
        else{
            cout<<front->data<<endl;
            front = front->next;
        }
    }

        void display(){
        if(rear == NULL){
            cout<<"Stack is Empty"<<endl;
        }
        else{
            Node* temp = front;
            while(temp != NULL){
                cout<<temp->data<<" ";
                temp = temp->next;
            }
            cout<<endl;
        }

    }
};


int main(){
    Queue q1;
    int value;
```

```
        int option;
        do{
                cout<<"ENTER 1 FOR ENQUEUE"<<endl;
                cout<<"ENTER 2 FOR DEQUEUE"<<endl;
                cout<<"ENTER 3 FOR PEEK"<<endl;
                cout<<"ENTER 4 FOR DISPLAY"<<endl;
                cout<<"ENTER 0 FOR EXIST"<<endl;
                cout<<"SELECT:"<<endl;
                cin>>option;
                if(option == 1){
                        cout<<"ENTER THE VALUE"<<endl;
                        cin>>value;
                        q1.Enqueue(value);
                }
                else if(option == 2){
                        q1.Dequeue();

                }
                else if(option == 3){
                        cout<<q1.peek()<<endl;

                }
                else if(option == 4){
                        q1.display();
                }
                else{
                        cout<<"Wrong Input Please Select from 1 - 4"<<endl;
                }

        }while(option != 0);


}
```

Picture:

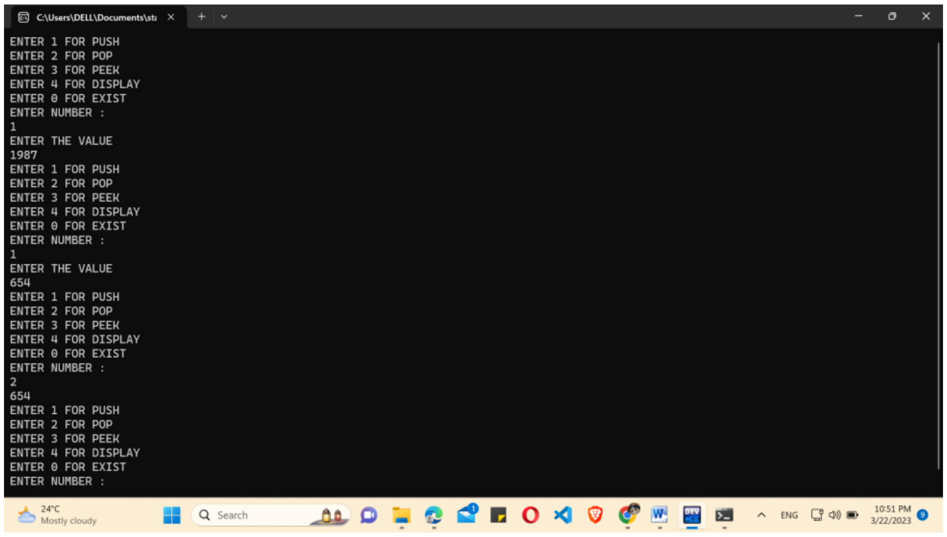


```
ENTER 1 FOR ENQUEUE
ENTER 2 FOR DEQUEUE
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
SELECT:
1
ENTER THE VALUE
190
ENTER 1 FOR ENQUEUE
ENTER 2 FOR DEQUEUE
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
SELECT:
1
ENTER THE VALUE
45
ENTER 1 FOR ENQUEUE
ENTER 2 FOR DEQUEUE
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
SELECT:
2
190
ENTER 1 FOR ENQUEUE
ENTER 2 FOR DEQUEUE
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
SELECT:
```

5. Stack using Linked list

Source code:

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

class Node {
        public:
                int data;
                Node* next;
        Node(int data){
                this->data = data;
                this->next = NULL;
        }

};


class stack{
        public:
                Node* Head;

        stack(){
                Head = NULL;
        }

        void push(int data){
                Node* newNode = new Node(data);
                newNode->next = Head;
                Head = newNode;
        }

        void Pop(){
                if(Head == NULL ){
                        cout<<"Stack is Empty"<<endl;
                }
                else{
                        Node* temp = Head;
                        cout<<temp->data<<endl;
                        Head = Head->next;

                        delete temp;
                }

        }

        int peek(){
                return  Head->data;
        }

        void display(){
                if(Head == NULL){
                        cout<<"Stack is Empty"<<endl;
                }
                else{
                        Node* temp = Head;
                        while(temp != NULL){
                                cout<<temp->data<<" ";
                                temp = temp->next;
                        }
                        cout<<endl;
                }

        }
};

int main(){
        stack s1;
        int value;
        int option;
        do{
                cout<<"ENTER 1 FOR PUSH"<<endl;
                cout<<"ENTER 2 FOR POP"<<endl;
                cout<<"ENTER 3 FOR PEEK"<<endl;
                cout<<"ENTER 4 FOR DISPLAY"<<endl;
                cout<<"ENTER 0 FOR EXIST"<<endl;
                cout<<"ENTER NUMBER :"<<endl;
                cin>>option;
                if(option == 1){
                        cout<<"ENTER THE VALUE"<<endl;
                        cin>>value;
                        s1.push(value);
                }
                else if(option == 2){
                        s1.Pop();

                }
                else if(option == 3){
                        cout<<s1.peek()<<endl;

                }
                else if(option == 4){
                        s1.display();
                }
                else{
                        cout<<"Wrong Input Please Select from 1 - 4"<<endl;
                }

        }while(option != 0);


}
```
Picture:



```
ENTER 1 FOR PUSH
ENTER 2 FOR POP
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
ENTER NUMBER :
1
ENTER THE VALUE
1987
ENTER 1 FOR PUSH
ENTER 2 FOR POP
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
ENTER NUMBER :
1
ENTER THE VALUE
654
ENTER 1 FOR PUSH
ENTER 2 FOR POP
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
ENTER NUMBER :
2
654
ENTER 1 FOR PUSH
ENTER 2 FOR POP
ENTER 3 FOR PEEK
ENTER 4 FOR DISPLAY
ENTER 0 FOR EXIST
ENTER NUMBER :
```