AIM: Implement the following stack operations such as push, pop, display, count number of elements in stack, top using Array and Linked List using a Menu driven Format.

THEORY:

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last In First Out) or FILO (First In Last Out).

Mainly the following three basic operations are performed in the stack:

- Push :- Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- Pop :- Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- Peek or Top :- Returns top element of stack.
- isEmpty :- Returns true if stack is empty, else false.
- Count: Returns number of elements in a stack.

There are two ways to implement a stack:

- **Using array:**

A stack data structure can be implemented using a one-dimensional array. But stack implemented using array stores only a fixed number of data values. This implementation is very simple. We have to define a one dimensional array of specific size and insert or delete the values into that array by using LIFO principle with the help of a variable called 'top'. Initially, the top is set to -1. Whenever we want to insert a value into the stack, increment the top value by one and then insert. Whenever we want to delete a value from the stack, then delete the top value and decrement the top value by one.

- **Using linked list:**

A stack can be easily implemented through the linked list. In stack Implementation, a stack contains a top pointer. which is the “head” of the stack where pushing and popping items happens at the head of the list. The first node has null in the link field and the second node link has the first node address in the link field and so on and the last node address in “top” pointer.

The main advantage of using linked lists over an array is that it is possible to implement a stack that can shrink or grow as much as needed. Using an array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocated. so overflow is not possible.

A) STACK USING ARRAY:

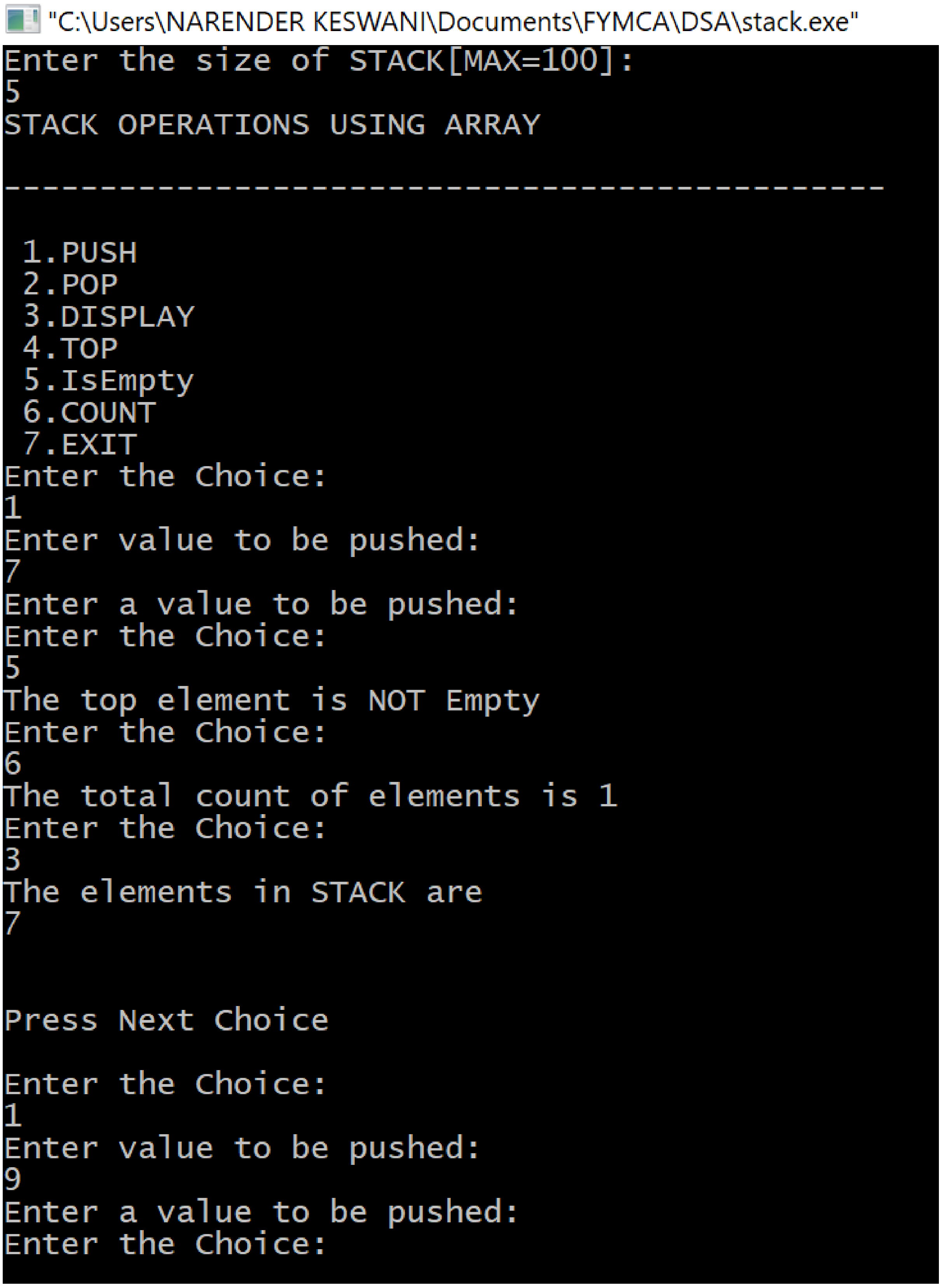
SOURCE CODE:

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

int stack[100],choice,n,top,x,i,val;

void push(int val)
{
   if(top>=n-1)
   {
      cout<<"Stack is Overflow"<<endl;
   }
   else
   {
      cout<<"Enter a value to be pushed:"<<endl;
      top++;
      stack[top] = val;
   }
}

void pop()
{
   if(top<=-1)
   {
      cout<<"Stack is Underflow"<<endl;
   }
   else
   {
      cout<<"The popped elements is"<<endl;
      cout<<stack[top]<<endl;
      top--;
   }
}
void display()
{
   if(top>=0)
   {
      cout<<"The elements in STACK are"<<endl;
      for(i=top; i>=0; i--)
         cout<<stack[i]<<endl;
      cout<<"\n"<<endl;
      cout<<"Press Next Choice \n"<<endl;
   }
   else
   {
      cout<<"The STACK is empty \n"<<endl;
   }
}
```

```
void Top()
{
   cout<<"The top element is "<<stack[top]<<"\n";
}

void IsEmpty()
{
   if(top == -1)
   {
      cout<<"The top element is Empty\n";
   }
   else
   {
      cout<<"The top element is NOT Empty\n";
   }
}

void countElements()
{
   cout<<"The total count of elements is "<<top+1<<"\n";
}

int main()
{
   top=-1;
   cout<<"Enter the size of STACK[MAX=100]:"<<endl;
   cin>>n;
   cout<<"STACK OPERATIONS USING ARRAY"<<endl;
   cout<<"\n---------------------------------------------"<<endl;
   cout<<"\n 1.PUSH\n 2.POP\n 3.DISPLAY\n 4.TOP\n 5.IsEmpty\n 6.COUNT\n 7.EXIT"<<endl;
   do
   {
      cout<<"Enter the Choice:"<<endl;
      cin>>choice;
      switch(choice)
      {
      case 1:
      {
         cout<<"Enter value to be pushed:"<<endl;
         cin>>val;
         push(val);
         break;
      }
      case 2:
      {
         pop();
         break;
      }
      case 3:
      {
         display();
```

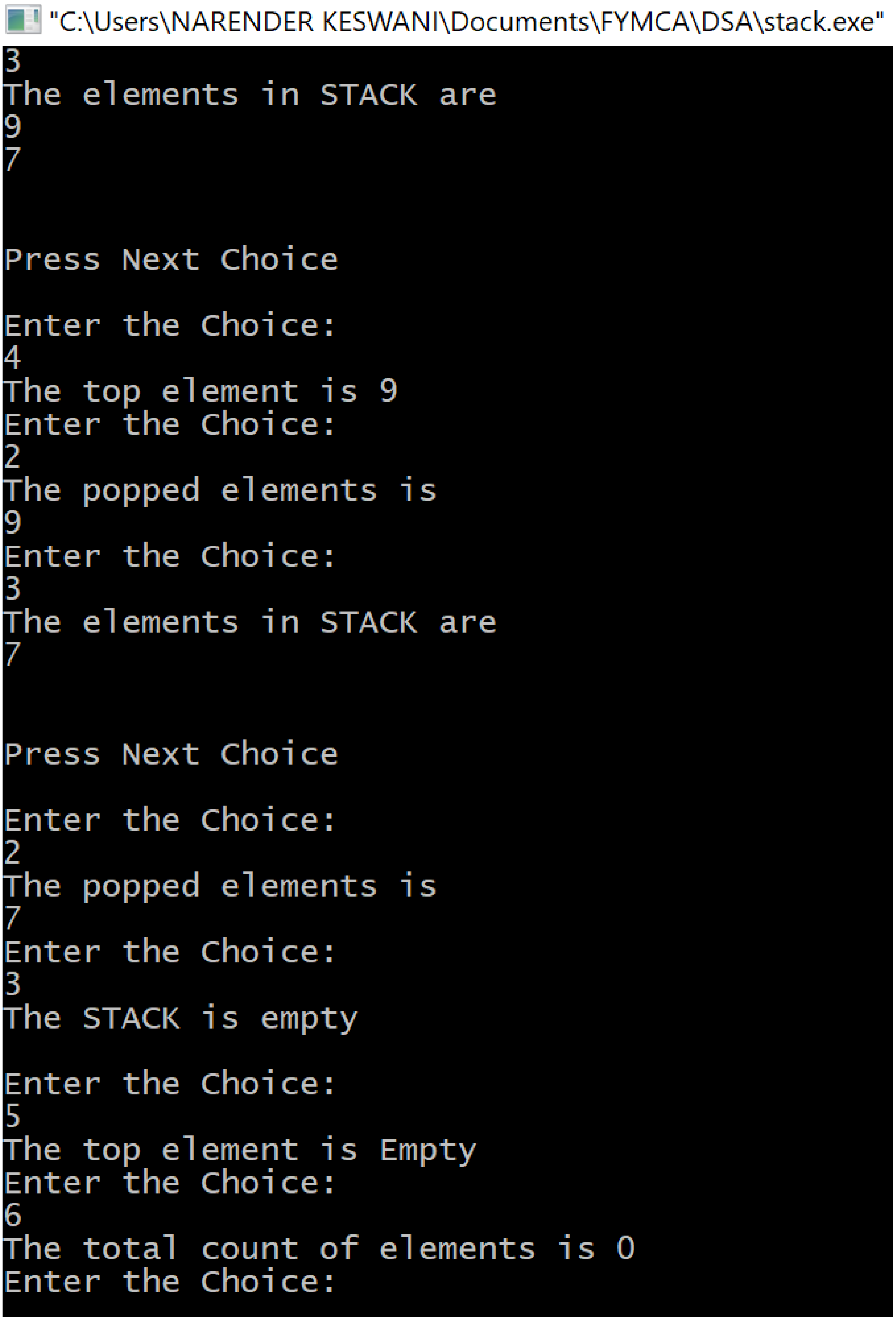
```
            break;
        }
        case 4:
        {
            Top();
            break;
        }
        case 5:
        {
            IsEmpty();
            break;
        }
        case 6:
        {
            countElements();
            break;
        }
        case 7:
        {
            cout<<"Exit \n"<<endl;
            break;
        }
        default:
        {
            cout<<"Invalid Input!, Please Enter a Valid Choice(1/2/3/4/5/6/7)"<<endl;
        }
        }
    }
    while(choice!=7);

    return 0;
}
```

OUTPUT:



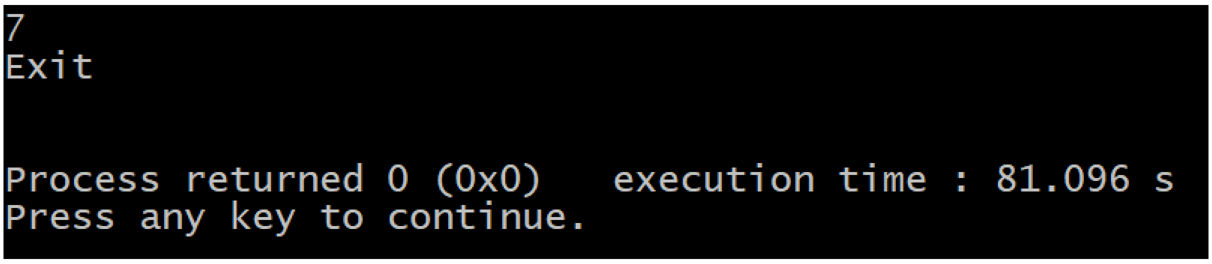
"C:\Users\NARENDER KESWANI\Documents\FYMCA\DSA\stack.exe"

```
Enter the size of STACK[MAX=100]:
5
STACK OPERATIONS USING ARRAY

---------------------------------------------


 1.PUSH
 2.POP
 3.DISPLAY
 4.TOP
 5.IsEmpty
 6.COUNT
 7.EXIT
Enter the Choice:
1
Enter value to be pushed:
7
Enter a value to be pushed:
Enter the Choice:
5
The top element is NOT Empty
Enter the Choice:
6
The total count of elements is 1
Enter the Choice:
3
The elements in STACK are
7


Press Next Choice

Enter the Choice:
1
Enter value to be pushed:
9
Enter a value to be pushed:
Enter the Choice:
```



"C:\Users\NARENDER KESWANI\Documents\FYMCA\DSA\stack.exe"

```
3
The elements in STACK are
9
7


Press Next Choice

Enter the Choice:
4
The top element is 9
Enter the Choice:
2
The popped elements is
9
Enter the Choice:
3
The elements in STACK are
7


Press Next Choice

Enter the Choice:
2
The popped elements is
7
Enter the Choice:
3
The STACK is empty

Enter the Choice:
5
The top element is Empty
Enter the Choice:
6
The total count of elements is 0
Enter the Choice:
```



```
7
Exit


Process returned 0 (0x0)   execution time : 81.096 s
Press any key to continue.
```

B) <u>STACK USING LINKED LIST:</u>

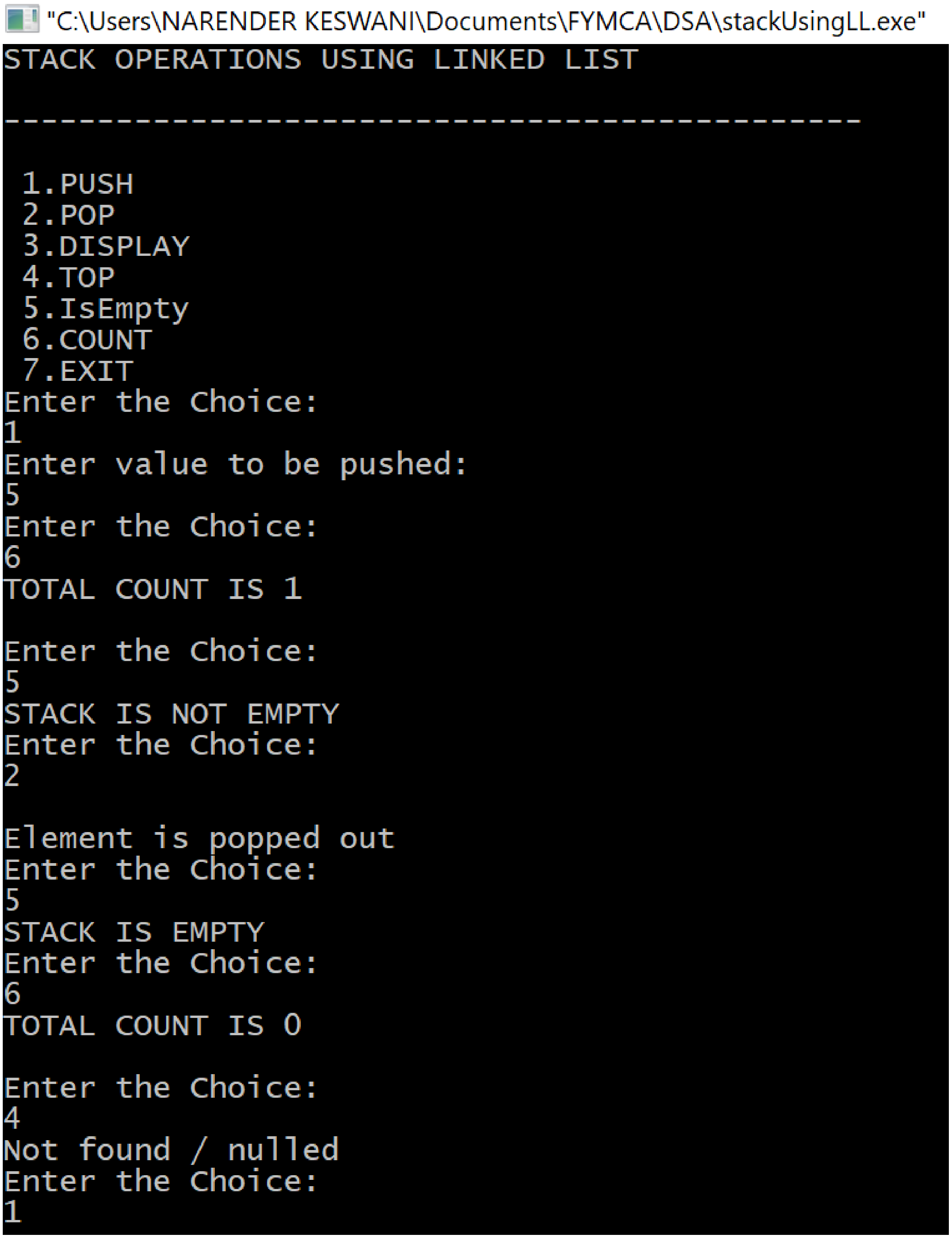
<u>SOURCE CODE:</u>

```
#include <bits/stdc++.h>
using namespace std;

struct Node
{
   int data;
   struct Node* link;
};

struct Node* top;

void push(int data)
{

   struct Node* temp;
   temp = new Node();

   if (!temp)
   {
      cout << "\nHeap Overflow";
      exit(1);
   }

   temp->data = data;
   temp->link = top;
   top = temp;
}

void IsEmpty()
{
   if(top == NULL)
   {
      cout<<"STACK IS EMPTY"<<endl;
   }
   else
   {
      cout<<"STACK IS NOT EMPTY"<<endl;
   }
}
```

```
int topElement()
{

   if (top != NULL)
   {
      cout<<"Top Element is "<<top->data<<" \n"<<endl;
   }
   else
   {
      cout<<"Not found / nulled"<<endl;
   }
}

void countElements()
{
   int count = 0;
   Node *temp = top;
   while(temp!=NULL)
   {
      count++;
      temp = temp->link;
   }
   cout<<"TOTAL COUNT IS "<<count<<"\n"<<endl;
}

void pop()
{
   struct Node* temp;

   if (top == NULL)
   {
      cout << "\nStack Underflow" << endl;
   }
   else
   {
      temp = top;
      top = top->link;
      temp->link = NULL;
      free(temp);
      cout << "\nElement is popped out" << endl;
   }
}

void display()
{
   struct Node* temp;

   if (top == NULL)
   {
      cout << "\nStack Underflow";
      exit(1);
   }
   else
   {
      temp = top;
      cout << "Elements are \n"<<endl;
      while (temp != NULL)
      {
         cout << temp->data<<"\n";
         temp = temp->link;
      }
   }
}

int main()
{
   int choice,val;
   cout<<"STACK OPERATIONS USING LINKED LIST"<<endl;
   cout<<"\n---------------------------------------------"<<endl;
   cout<<"\n 1.PUSH\n 2.POP\n 3.DISPLAY\n 4.TOP\n 5.IsEmpty\n 6.COUNT\n 7.EXIT"<<endl;
   do
   {
      cout<<"Enter the Choice:"<<endl;
      cin>>choice;
      switch(choice)
      {
      case 1:
      {
         cout<<"Enter value to be pushed:"<<endl;
         cin>>val;
         push(val);
         break;
      }
      case 2:
      {
         pop();
         break;
      }
      case 3:
      {
         display();
         break;
      }
      case 4:
      {
         topElement();
         break;
      }
      case 5:
      {
         IsEmpty();
```

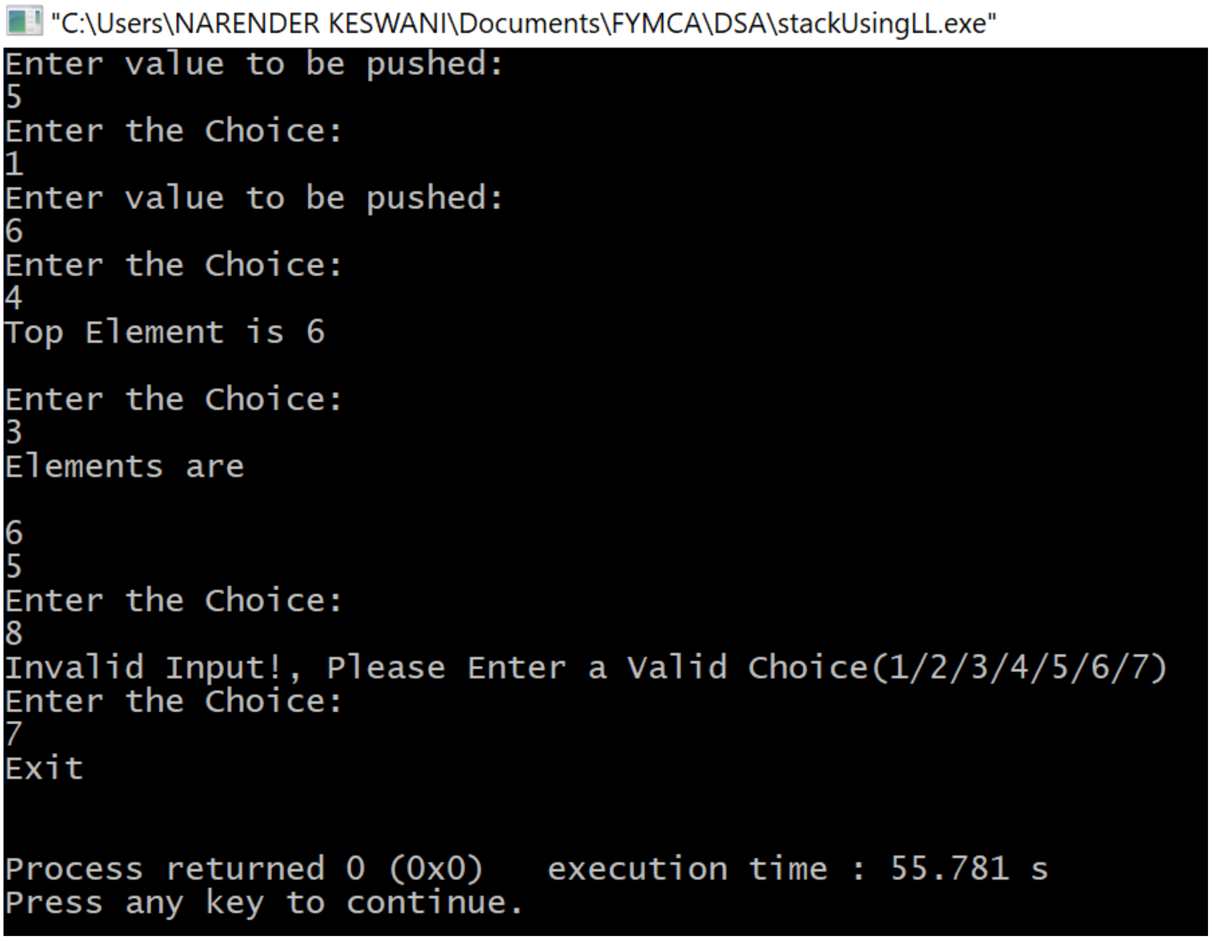
```
        break;
      }
      case 6:
      {
        countElements();
        break;
      }
      case 7:
      {
        cout<<"Exit \n"<<endl;
        break;
      }
      default:
      {
        cout<<"Invalid Input!, Please Enter a Valid Choice(1/2/3/4/5/6/7)"<<endl;
      }
      }
  }
  while(choice!=7);
  return 0;
}
```

OUTPUT:



"C:\Users\NARENDER KESWANI\Documents\FYMCA\DSA\stackUsingLL.exe"

```
STACK OPERATIONS USING LINKED LIST

----------------------------------------------

 1.PUSH
 2.POP
 3.DISPLAY
 4.TOP
 5.IsEmpty
 6.COUNT
 7.EXIT
Enter the Choice:
1
Enter value to be pushed:
5
Enter the Choice:
6
TOTAL COUNT IS 1

Enter the Choice:
5
STACK IS NOT EMPTY
Enter the Choice:
2

Element is popped out
Enter the Choice:
5
STACK IS EMPTY
Enter the Choice:
6
TOTAL COUNT IS 0

Enter the Choice:
4
Not found / nulled
Enter the Choice:
1
```


"C:\Users\NARENDER KESWANI\Documents\FYMCA\DSA\stackUsingLL.exe"

```
Enter value to be pushed:
5
Enter the Choice:
1
Enter value to be pushed:
6
Enter the Choice:
4
Top Element is 6

Enter the Choice:
3
Elements are

6
5
Enter the Choice:
8
Invalid Input!, Please Enter a Valid Choice(1/2/3/4/5/6/7)
Enter the Choice:
7
Exit


Process returned 0 (0x0)   execution time : 55.781 s
Press any key to continue.
```

CONCLUSION:

I have learned about stacks and their implementation in arrays and linked lists. The main advantage of using linked lists over an array is that it is possible to implement a stack that can shrink or grow as much as needed. Using an array will put a restriction to the maximum capacity of the array which can lead to stack overflow. Here each new node will be dynamically allocated. so overflow is not possible.