# Jaskirat Singh

# 2020CSC1008

## STACK USING LINKED LIST

```
#include <iostream>
#include<cstring>

using namespace std;

struct Node{
   char data;
   struct Node * next;
}*top=NULL;

void Push(char x)
{   struct Node* p=new Node;
   if(p==NULL)
      cout<<"\nStack Overflow";
   else{
      p->data=x;
      p->next=top;
      top=p;
   }
}

char Pop(){
   char x=-1;
   struct Node * p;
   if(top==NULL)
      cout<<"\nStack is Empty";
```

```
    else{
        p=top;
        x=p->data;
        top = top->next;
        delete p;
    }
    return x;
}

int isEmpty(){
    return top?0:1;
}

int Pre(char ch)
{


  if(ch == '-' || ch == '+') return 1;
  if(ch == '*' || ch == '/') return 2;
  else
  return 0;
}

char * InToPost(char *infix){
    char *postfix = new char[strlen(infix)+1];
    int i=0,j=0;
    while(infix[i]!='\0')
    {
        if(Pre(infix[i])==0)   // if char is operand then push into postfix array

        postfix[j++] = infix[i++];   // you were not using j++ with postfix array
```

```
    else{

      if(isEmpty()==1)  // if stack is empty then push into Stack

         Push(infix[i++]);
      else
      {
         if(Pre(infix[i])>=Pre(top->data))  // If coming operator has same or less predence then push into Stack
            Push(infix[i++]);
         else{
            while(Pre(infix[i])<=Pre(top->data))
              postfix[j++]= Pop();
         }
       }
     }
   }
   while(!isEmpty())
      postfix[j++]=Pop();
   postfix[j] = '\0';
   return postfix;
}

int input()
{
   cout<<" 1. To push element in stack. \n 2. To pop first element from stack \n 3. To change expression \n 4. If you want to exit. \n ";
   int choice;
   cout<<"Enter: ";
   cin>>choice;
   cout<<"\n";
```

```
    return choice;
}

int main(){
    int choice= input();
    while(choice!=4)
    {
        switch(choice)
        {
            case 1:
            {
              char x;
              cout<<" Enter element to push: ";
              cin>>x;
              Push(x);
              choice= input();
              break;
            }
            case 2:
            {
               cout << Pop() << " popped from stack\n";
               choice= input();
               break;
            }
            case 3:
            {
                cout<<"Enter expression to convert: ";
                char *infix = new char[30];
                cin>>infix;
                cout<<"\n The changed expression is: "<<InToPost(infix)<<endl;
                choice= input();
```

```
            break;
        }
        case 4:
        {
          choice= input();
          break;
        }
        default:
        cout<<"Wrong choice!";
      }
   }

}
```

```
 Enter: 1

 Enter element to push: d
 1. To push element in stack.
 2. To pop first element from stack
 3. To change expression
 4. If you want to exit.
 Enter: 1

 Enter element to push: u
 1. To push element in stack.
 2. To pop first element from stack
 3. To change expression
 4. If you want to exit.
 Enter: 2

u popped from stack
 1. To push element in stack.
 2. To pop first element from stack
 3. To change expression
 4. If you want to exit.
 Enter: 3

Enter expression to convert: b*c+d*f

 The changed expression is: bc*df*+d
 1. To push element in stack.
 2. To pop first element from stack
 3. To change expression
 4. If you want to exit.
 Enter: 2


Stack is Empty popped from stack
 1. To push element in stack.
 2. To pop first element from stack
 3. To change expression
 4. If you want to exit.
 Enter:
```