# Experiment No.2 : STACK

Name: Shubham Shatrughna Shendage
Class:SE
Branch:IT
Batch:D
Roll No:561

```
#include <iostream>

#include<string.h>

using namespace std;


class node

{

   public:

   char d;

   node*next;

};

class stack

{

   node*top;

   public:

   stack()

   {

      top=NULL;
```

```
 }
   void push(char);
   char pop();
   void dis();
   int emp();
};
float Operation(char Op,float A, float B)
{
   int I=0;
   float P=1;
   if(Op=='*')
      P=A*B;
   else if(Op=='/')
      P=A/B;
   else if(Op=='+')
      P=A+B;
   else if (Op=='-')
      P=A-B;
   else while(I++<B)
      P=P*A;
   return P;

}

int Priority(char Op)
{
    if (Op=='^')
        return 2;
    if(Op=='+'|| Op=='-')
        return 0;
    else
        return 1;
}
char stack::pop()
{
    if (emp()==1)
    {
        cout<<"\nUnderflow";
        return -1;
  }
    else
    {
        node *p=top;
        top=top->next;
        char x= p->d;
        delete p;
        return x;
    }
}

void stack::push(char d1)
{
    node*p=new node;
    p->d=d1;
    p->next=top;
    top=p;
}
void stack::dis()
{
    node*p=top;
    while(p!=NULL)
    {
        cout<<p->d<<"\n";
        p=p->next;
    }
}
int stack::emp()
{
    if(top==NULL)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
void infix_to_postfix(char String[])
```

```
{
   char PostExpression[25],opr;
   int I=0,J=0;
   stack s;
   for(I=0;I<strlen(String);I++)
   {
      if(isalnum(String[I]))
            PostExpression[J++]=String[I];
      else
      {
         if(String[I]==')')
            {
            opr=s.pop();
            while(opr!='(')
               {
               PostExpression[J++]=opr;
               opr=s.pop();
               }
            }
         else
            {
            if (String[I]=='(')
               s.push(String[I]);
            else
               {
               while(!s.emp())
```

```
                {
                opr=s.pop();
                if(opr!='('&&Priority(opr)>=Priority(String[I]))
                {
                   PostExpression[J++]=opr;
                }
                else
                {
                   s.push(opr);
                   break;
                }
                }//while
             s.push(String[I]);
             }
          }//else
       }
   }//for
   while(!s.emp())
   {
      PostExpression[J++]=s.pop();
     PostExpression[J]='\0';
     cout<<"\nPost: "<<PostExpression;
   }
}
void InfixToPrefix(char String[])
{
```

```
char PreExpression[20],opr;

int I=0,J=0;

I=strlen(String); // @suppress("Function cannot be resolved")

I--;

stack s;

for(I=strlen(String);I>=0;I--)

{

   if(isalnum(String[I]))

   {

      PreExpression[J++]=String[I];

   }else

   {

      if(String[I]=='(')

      {

         opr=s.pop();

         while(opr!=')')

         {

            PreExpression[J++]=opr;

            opr=s.pop();

         }

      }

      else

      {

         if (String[I]==')')

            s.push(String[I]);

         else
```

```
            {
               while(!s.emp())
               {
                  opr=s.pop();
                  if(opr!=')'&&Priority(opr)>=Priority(String[I]))
                  {
                     PreExpression[J++]=opr;
                        }
                  else
                     {
                     s.push(opr);
                     break;
                     }
               }//while
               s.push(String[I]);
            }
         }//else
      }
   }//for
   //while(!s.emp())
// {
//     PreExpression[J++]=s.pop();
//  PreExpression[J]='\0';
//  cout<<"\nPre: "<<PreExpression;
   //for(I=J;I>=0;I--){
   //cout<<PreExpression[I];
```

```
  //}
// }
  while (!s.emp()) {
      PreExpression[J++] = s.pop();
    }
    PreExpression[J] = '\0';
    // Reverse the expression
    cout << "Prefix Expression: ";
    for (I = J - 1; I >= 0; I--) {
      cout << PreExpression[I];
    }
    cout << "\n";
}
int main()
{
  //stack s;
  int ch1;
  //char ch;
  char Infix_expression[100];
 // char expression[100];

  //Infix_expression[100]==NULL;
  //expression[100]==NULL;

  do
  {
```

```
cout<<"\n1:Infix to postfix";

cout<<"\n2:Infix to Prefix";

cout<<"\n3:Exit";

cout<<"\nEnter your choice \t";

cin>>ch1;


switch(ch1)

{

case 1:

  cout<<"\n\n Enter the Infix Expression:";

  cin>>Infix_expression;

  infix_to_postfix(Infix_expression);

  break;


case 2:

  cout<<"\n\n Enter the Infix Expression:";

  cin>>Infix_expression;

  InfixToPrefix(Infix_expression);

  break;

case 3:

  break;

}


}

while(ch1!=4);

return(0);}
```

## Output:

```
1:Infix to postfix

2:Infix to Prefix

3:Exit

Enter your choice 1



 Enter the Infix Expression:A+B


Post: AB+

1:Infix to postfix

2:Infix to Prefix

3:Exit

Enter your choice 2



 Enter the Infix Expression:A-B

Prefix Expression: -A .B

1:Infix to postfix

2:Infix to Prefix

3:Exit

Enter your choice
```