Name: Sakshi Mishra

Roll No:53

Subject: DSA

Assignment No. 02

```
#include<iostream>
#include<ctype.h> //it is included for using function ..isalnum()
#include<string.h>
#include<math.h> using
namespace std; struct
node
{
 char data;  struct
node *next;
};
class stack
{
node *top;
public :
stack()
{
top=NULL;
}
 char Top()
 {
 return (top->data);
}
 void push(char x)
 {
 node         *temp;
temp=new   node;
temp->data=x;
temp->next=top;
top=temp;
}
char pop()
 {
 char value;  value=top-
>data; top=top->next;
return(value);
}
int isempty()
 {

if(top==NUL
L)  return 1;
else  return 0;
}
};
int priority(char op)
{
if(op=='(' || op==')') return 0; else
if(op=='+' || op=='-') return 1; else
if(op=='*' || op=='/' || op=='%') return
2; else if(op=='^') return 3; else
return 4;
}
int operation(char op,int A,int B)
{
if(op=='*')
return A*B; else
if(op=='/') return
A/B; else
if(op=='^')
```

```
return
pow(A,B);
else if(op=='+') return
A+B;
else  if(op=='-
') return A-B;
else return -1;
}
void infixtopostfix(char infix[50]) // (a+b)*c infix expre...it is string
{
 char token, operand, post[50]; // token= will read all characters from given expression  int
i, j=0; //operand=a, b, c // post[50] will stored our output
 stack S;  for(i=0; infix[i]!='\0'; i++) // i=0 1
2 3 4 5 6 7
{ // ( a + b ) * c '\0'
 token=infix[i]; // when i=2, token=infix[2], token=+
if(isalnum(token)) //it will check the token is alphabet or number
post[j++]=token; //post[]= a  else  if(token=='(') //this will get
execute
 S.push(token); // ( ... it will be pushed into stack
else   if(token==')')
while((operand=S.pop())!='(')
post[j++]=operand;
 else
 {
 while(!S.isempty() && priority(S.Top())>=priority(token))
post[j++]=S.pop();
 S.push(token);
}
}
while(!S.isempty()) post[j++]=S.pop(); // ab+c*
post[j]='\0'; //this will indicate end of the string
cout<<post;
}
void infixtoprefix(char infix[50])
{
 char token, operand, pre[50];
int i, j=0;  stack S;
for(i=strlen(infix)-1; i>=0; i-
-)
{
 token=infix[i];
if(isalnum(token))
pre[j++]=token;  else
if(token==')')  S.push(token);
else if(token=='(')
while((operand=S.pop())!=')')
pre[j++]=operand;  else
 {
while(!S.isempty() && priority(S.Top())>priority(token)) pre[j++]=S.pop();
 S.push(token);
}
}
while(!S.isempty())
pre[j++]=S.pop();
pre[j]='\0';  //Displaying in
reverse for(i=strlen(pre)-1;
i>=0; i--)  cout<<pre[i];
}
float postfixevaluation(char exp[50])
{
int i,val;
```

```
char token; float
Operand1,Operand2,Result;
stack S; for(i=0;exp[i]!='\0';i++)
{
token=exp[i];
if(isdigit(token))
{
S.push(token-48);
}
else
{
 Operand2=S.pop();
 Operand1=S.pop();
 Result=operation(token,Operand1,Operand2);
 S.push(Result);
}
}
return S.pop();
}
float prefixevaluation(char Str[50])
{
int i,val;
float Op1,Op2,Result;
stack S; for(i=strlen(Str)-1;i>=0;i-
-)
{
 if(isdigit(Str[i]))
 {
 S.push(Str[i]-48);
 }
 else
 {
```

```
Op1=S.pop();
Op2=S.pop();
Result=operation(Str[i],Op1,Op2);
S.push(Result);
 }
}
return S.pop();
}
int main()
{
int choice; char expression[50]; // Delaring character array to enter
expression (a+b)*c do
{
cout<<"\nEnter Choice of Operation:\n 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation
4. Prefix Evaluation 5. Exit\n"; cin>>choice; switch(choice)
{
case 1: cout<<"Enter Infix Expression\n";
cin>>expression; // (a+b)*c infixtopostfix(expression);
//function will get called break; case 2: cout<<"Enter Infix
Expression\n"; cin>>expression; infixtoprefix(expression);
break; case 3: cout<<"Enter postfix Expression\n";
cin>>expression;
cout<<"Answer:\n"<<postfixevaluation(expression)<<endl
; break; case 4: cout<<"Enter prefix Expression\n";
cin>>expression;
cout<<"Answer:\n"<<prefixevaluation(expression)<<endl;
break; case 5: cout<<"End of program\n";  break; default :
cout<<"Wrong Choice\n";  break;
}
}while(choice!=5);
}
```

*********************************Output********************************

```
Enter Choice of Operation:
 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation 4. Prefix Evaluation 5. Exit
1
Enter Infix Expression
(6+7)*(1+2)
67+12+*
Enter Choice of Operation:
 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation 4. Prefix Evaluation 5. Exit
2
Enter Infix Expression
(6+7)*(1+2)
*+67+12
Enter Choice of Operation:
 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation 4. Prefix Evaluation 5. Exit
3
Enter postfix Expression
5432+-*
Answer:
-5

Enter Choice of Operation:
 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation 4. Prefix Evaluation 5. Exit
4
Enter prefix Expression
+-*4567
Answer:
21

Enter Choice of Operation:
 1. Infix to Postfix 2. Infix to Prefix 3. Postfix Evaluation 4. Prefix Evaluation 5. Exit
5
End of program
```