**Experiment No.9**

**Aim:** Implementation of infix to postfix conversion and evaluation of postfix expression.

**Code:**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define max 20

struct stack //strucutre of stack to store operators

{

int top;

char stack[max];//stack stores string values

}s;

void push(char x)//push function to push operators on stack

{

if(s.top>=max-1)//check if stack is full

printf("\n\nOVERFLOW ! ! !");

else //if stack is not empty push the operator

{

s.top++;

s.stack[s.top]=x;

}

}

char pop()//pop function to pop operators from stack

{

char x;

if(s.top==(-1))

printf("\n\nUNDERFLOW ! ! !");

else

{

x=s.stack[s.top];

s.top--;

}

return x;//return the top most element from stack

}

char stacktop() //return tpye changed(int->char)

{

if(s.top==-1)

return -1;

else

return s.stack[s.top];

}

int isoperand(char c) //check if current element is operand i.e. in between A to Z

{

if((c>='A' && c<='Z')||(c>='a' && c<='z'))

return 1;

else

return 0;

}

int op(char x)// assign prorities to operators

{

int r;

switch(x)

{

case '(' :r=1;

break;

case '+' :

case '-' :r=2;

break;

case '\*' :

case '/' :r=3;

break;

case '^' :r=4;

break;

}

return r;

}

int prcd(char a,char b)//compare priorities of operators

{

if(op(a)>=op(b))

return 1;

else

return 0;

}

void main()

{

char ip[40],out[40],temp;

int i,j,a,x,op1,op2,result;

s.top=-1; //set top of stack to -1

printf("\nEnter infix expression : ");

scanf("%s",ip);

for(i=0,j=0;ip[i]!='\0';i++)

{

if(ip[i]=='(') //element is "(" then push the element.

{

push(ip[i]);

}

else

if(ip[i]==')')//element is ")" then pop the elements till ")" or stack isempty

{

while(s.top!=-1 && s.stack[s.top]!='(')

{

out[j++]=pop();

}

if(s.top==-1)// if we provide only ")" then it is an incorrect expression

{

printf("\nINCORRECT EXPRESSION");

break;

}

temp=pop();

}

else //element is an operand then no need to push just store in out[]

if(isoperand(ip[i]))

{

out[j++]=ip[i];

//i++;

}

else //element is an operator

{

a=prcd(stacktop(),ip[i]);//get precedence of the operator and compare with

//precedence of operator present at stack top

if(a==0)

push(ip[i]);//if the precedence is lower push the operator

else //if the precedence is higer then pop the operators till stack is empty //or we get an

opertor having precedence higher

{

while(stacktop()!=-1 && prcd(stacktop(),ip[i])!=0)

out[j++]=pop();

push(ip[i]);

}//end of else

}//end of else

}//end of for loop

while(s.top!=-1)/\*store all the remaining elements from stack to out[]\*/

out[j++]=pop();/\*and make stack empty\*/

out[j]='\0';//put "/0" at the end of the string

printf("\nPost-fix Expression : %s\n",out);

int n=strlen(out);

printf("\n Evalution of the expression\n");

for(i=0;i<n;i++) //ab+c\*

{

if(isalpha(out[i]))//if element is alphabet

{

printf("\nEnter a value for the variable %c :",out[i]);

scanf("%d",&x);

push(x);//push the value of var

}

else //if element is operator

{

op2=pop();//pop value of 1st var

op1=pop();//pop value of 2nd var

switch(out[i])

{

case '+': result=op1+op2;

break;

case '-': result=op1-op2;

break;

case '\*' :result=op1\*op2;

break;

case '/': result=op1/op2;

break;

case '^': result=op1^op2;

break;

}

push(result);//store result in stack

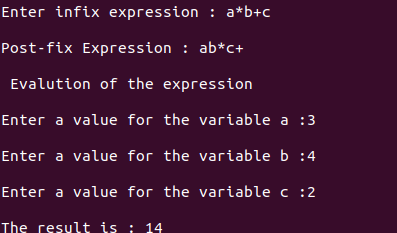
}

}

printf("\nThe result is : %d\n",result);

}

**Output:**



Conclusion: Thus we can convert the given infix expression into postfix and evaluate postfix expression using STACK