**LAB 2**

**RANJAN DEVI 1BM22CS219**

**1.) Infix to Postfix conversion**

#include <stdio.h>

#include <string.h>

int i,j=0,n,top=-1;

char val,temp,c;

char stack[100];

char infix[100],postfix[100];

void push(char );

char pop();

int precedence( char);

void main()

{

printf("Enter an infix expression:\n");

scanf("%s",infix);

n=strlen(infix);

for(i=0;i<=n;i++)

{

if(infix[i]=='(')

push(infix[i]);

else if(infix[i]>='a' && infix[i]<='z' || infix[i]>='A' && infix[i]<='Z' || infix[i]>='0' && infix[i] <='9')

{

postfix[j]=infix[i];

j++;

}

else if(infix[i]==')')

{

while(stack[top]!='(')

{ postfix[j]=pop();

j++;

}

temp=pop();

}

else

{

while(precedence(infix[i])<=precedence(stack[top]) && top>=0)

{

postfix[j]=pop();

j++;

}

push(infix[i]);

}

}

while(stack[top]!='\0')

{

postfix[j]=pop();

j++;

}

postfix[j]='\0';

printf("Postfix expression :\n");

printf("%s",postfix);

}

void push(char val)

{

if(top==n-1)

printf("Stack is full");

else

{

top+=1;

stack[top]=val;

}

}

char pop()

{

if(top==-1)

printf("Stack is empty ");

else

{

val=stack[top];

top-=1;

return val;

}

}

int precedence(char c)

{

if(c=='^')

return 5;

else if(c=='/')

return 4;

else if(c=='\*')

return 3;

else if(c=='+')

return 2;

else if(c=='-')

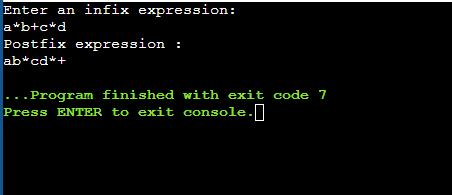
return 1;

else

return -1;

}

**OUTPUT:**



**2.) Evaluation of postfix expression**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int n,top=-1,k;

int val,res,ans;

int stack[100];

char expn[100];

void push(int [], int );

int pop(int []);

int evaluate(char []);

void main()

{

printf("Enter a postfix expression:\n");

scanf("%s",expn);

ans=evaluate(expn);

printf("Answer of above expression:\n");

printf("%d",ans);

}

int evaluate(char expn[])

{

int op1,op2,i=0;

n=strlen(expn);

while(expn[i] != '\0')

{

if (isdigit(expn[i]))

push( stack , (int)(expn[i] - '0'));

else

{

op1=pop(stack);

op2=pop(stack);

switch(expn[i])

{

case '+':

res=op1+op2;

break;

case '-':

res=op1-op2;

break;

case '\*':

res=op1\*op2;

break;

case '/':

res=op2/op1;

break;

case '%':

res=op2%op1;

break;

default:

continue;

}

if(res<0)

res=res\*(-1);

push(stack ,res);

}

i++;

}

return stack[top];

}

void push(int stack[] ,int val)

{

if(top==n-1)

printf("Stack is full");

else

{

top+=1;

stack[top]=val;

}

}

int pop(int stack[])

{

if(top==-1)

printf("Stack is empty ");

else

{

val=stack[top];

top-=1;

return val;

}

}

**OUTPUT:**

