Ex.No:4.2 Conversion of Infix to Postfix Expression using Stack ADT

Program:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define Blank ' '
#define Tab '\t'
#define MAX 20

int stack[MAX];
int top;
char infix[MAX],postfix[MAX];
void push(int);
int pop();
int prec(char);
int white_space(char);
void infix_to_postfix();
int eval_post();

void main()
{
int value;
char choice='y';
clrscr();
while(choice=='y')
{
 top = 0;
 printf("Enter infix : ");
 fflush(stdin);//clears previous input values//
 gets(infix);
 infix_to_postfix();
 printf("Postfix : %s\n",postfix);
 value=eval_post();
 printf("Value of expression : %d\n",value);
 printf("Do you wants to continue(y/n): ");
 scanf("%c",&choice);
 getch();
}
}

void infix_to_postfix()
{
int i,p=0,type,precedence,len;
char next ;
stack[top]='#'; //'#' represents NULL
len=strlen(infix);
infix[len]='#';
for(i=0;infix[i]!='#';i++)
```

```
{
 if(!white_space(infix[i]))
 {
 switch(infix[i])
 {
  case '(':
   push(infix[i]);
   break;
  case ')':
   while((next=pop())!='(')
    postfix[p++]=next;
   break;
  case '+':
  case '-':
  case '*':
  case '/':
  case '%':
  case '^':
   precedence = prec(infix[i]);
   while(stack[top]!='#' && precedence<=prec(stack[top]))
    postfix[p++]=pop();
   push(infix[i]);
   break;
  default: /*if an operand comes */
   postfix[p++] = infix[i];
  }
 }
}
while(stack[top]!='#')
 postfix[p++] = pop();
postfix[p] = '\0' ; /*End postfix with'\0' to make it a string*/
}

/* This function returns the precedence of the operator */
int prec(char symbol)
{
 switch(symbol)
 {
  case '(':
   return 0;
  case '+':
  case '-':
   return 1;
  case '*':
  case '/':
  case '%':
   return 2;
  case '^':
   return 3;
 }
}
```

```
void push(int elt)
{
 if(top>MAX)
 {
  printf("Stack overflow\n");
  exit(1);
 }
 else
 {
  top=top+1;
  stack[top] = elt;
 }
}

int pop()
{
 if(top==-1)
 {
  printf("Stack underflow \n");
  return;
 }
 else
  return(stack[top--]);
}

int white_space(char symbol)
{
 if(symbol==Blank||symbol==Tab||symbol=='\0')
  return 1;
 else
  return 0;
}

int eval_post()
{
int a,b,i,temp,result,len;
len=strlen(postfix);
postfix[len]='#';
for(i=0;postfix[i]!='#';i++)
{
 if(postfix[i]<='9' && postfix[i]>='0')
  push( postfix[i]-48 );
 else
 {
  a=pop();
  b=pop();
  switch(postfix[i])
  {
  case '+':
   temp=b+a; break;
```

```
  case '-':
   temp=b-a;break;
  case '*':
   temp=b*a;break;
  case '/':
   temp=b/a;break;
  case '%':
   temp=b%a;break;
  case '^':
   temp=pow(b,a);
  }
  push(temp);
 }
}
result=pop();
return result;
}
```

Output

```
Enter infix : (3+(6*9))/(9-(1*6))
Postfix : 369*+916*-/
Value of expression : 19
Want to continue(y/n) : y
Enter infix : (4+5)/(6-3)
Postfix : 45+63-/
Value of expression : 3
Want to continue(y/n) : n
```