// Program to implement Infix to postfix and postfix evaluation

#include<stdio.h>

#include<string.h>

#include<math.h> //for pow function

#include <ctype.h> //for isalnum function

char stack[50];

int top =-1,i;

int ptop=0;

char postf[50];

void pushpost(char ch)

{

  postf[ptop++] = ch; //pushing element to postfix output

}

int priority(char ch)

{ //Highest priority to 3

  if(ch=='(')

    return 0;

  if(ch=='+'|| ch=='-')

    return 1;

  if(ch=='/'|| ch== '\*')

    return 2;

  if(ch=='^')

    return 3;

}

void push(char ch)

{

  top ++;

  stack[top] = ch; // element inserted to stack

}

char pop()

{

if(top==-1)

     return -1;

   else

{

   return stack[top--]; //returns value after popped element

   }

}

void posteval()

{

top =-1; //clearing old stack

char b;

  int x,y,val=0;

  for(i=0;i<ptop;i++) //eval from left to right

  {

    b=postf[i];

    if(isdigit(b))

    push(b-'0'); //changing from ASCII value to integer

   else if(b=='+' || b == '-' || b == '/' || b=='\*' || b=='^')

       { x=pop();

       y=pop();

         switch(b)

         {

           case '+' : val = x+y;

                 break;

           case '-' : val = y-x;

                 break;

case '/' : val = y/x;

                break;

           case '\*' : val = y\*x;

                break;

           case '^' : val = pow(y,x);

           break;

         }

         push(val);

       }

}

  printf("The Postfix Evaluation Gives : %d \n", val);

}

void main()

{

  char expr[100],c,a;

int len;  //where a stores the returned value after popped element & c store each element of expression

  printf("Enter the expression\n");

  gets(expr);

   len = strlen(expr);

  for(i=0;i<len;i++)

  {

c= expr[i];

    if(isalnum(c))

       pushpost(c); //pushing to post expression

else

{

if( c=='(')

       push(c);

     else

     {

     if(c==')') //pops elements from stack till

     {

       a = pop(); // pops one element from stack and stores in a.

       while(a!='(')

       {

         pushpost(a); // popped element is pushed on to the postfix expression

         a= pop();

       }

     }

else

{

while(priority(stack[top])>= priority(c)) // pops only if operator inside stack has greater than or equal to priority than incoming element

        {

       a=pop();

        pushpost(a);

       }

       push(c);

     }

     }

}

  }

  //pushing all the remaining elements present in stack to postfix expression

  while(top!=-1)

  {

    a=pop();

    pushpost(a);

  }

  //printing postfix expression

printf("The Postfix Expression Is:\n");

  for(i=0;i<ptop;i++)

    printf("%c ",postf[i]);

  printf("\n");

posteval();

}

//End of main

**Output:**

