**Design, develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, \*, /, % (Remainder), ^ (Power) and alphanumeric operands.**

#include<stdio.h>

#include<ctype.h>

#include<math.h>

#include<string.h>

**// main program**

void main()

{

char infix[100],post[100],s[10],ch;

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

clrscr();

printf(" Enter the valid infix expression \n");

scanf("%s",infix);

push(s,&top, '#');

for(i=0;i<strlen(infix);i++)

{

ch = infix[i];

while(IPV(ch) < SPV(s[top]) )

post[j++] = pop(s, &top);

if(IPV(ch) > SPV(s[top]) )

push(s, &top, ch);

else

pop(s, &top);

}

while(s[top] != '#')

post[j++] = pop(s, &top);

post[j] = '\0';

printf(" The Converted postfix expression is %s \n", post);

}

**// Function to Insert Item into stack**

push(char s[100], int \*top, char item)

{

\*top = \*top + 1;

s[\*top]=item;

}

**// Function to delete item from stack**

pop(char s[100], int \*top)

{

int item;

item = s[\*top];

\*top = \*top - 1;

return item;

}

**// Input precedence value of a character**

IPV( char ch)

{

if(ch == '+' || ch == '-') return 1;

if(ch == '\*' || ch == '/') return 3;

if(ch == '(') return 9;

if(ch == ')') return 0;

if(ch == '^') return 6;

return 7;

}

**// Stack precedence value of a character**

SPV( char ch)

{

if(ch == '+' || ch == '-') return 2;

if(ch == '\*' || ch == '/') return 4;

if(ch == '(') return 0;

if(ch == '^') return 5;

if(ch == '#') return -1;

return 8;

}

**OUTPUT**

Enter the valid infix expression

(3^2\*5)/(3\*2-3)+5

The Converted postfix expression is 32^5\*32\*3-/5+

Enter the valid infix expression

(A+(B\*C-(D/E^F)\*G)\*H)

The Converted postfix expression is ABC\*DEF^/G\*-H\*+

Enter the valid infix expression

a+b\*c

The Converted postfix expression is abc\*+

Enter the valid infix expression

(a+b)\*c+(d-a)

The Converted postfix expression is ab+c\*da-+