**4. Develop 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<stdlib.h>

void evaluate();

void push(char);

char pop();

int prec(char);

char infix[30], postfix[30], stack[30];

int top = -1;

void main()

{

            printf("\nEnter the valid infix expression:\t");

            scanf("%s", infix);

            evaluate();

            printf("\nThe entered infix expression is :\n %s \n", infix);

            printf("\nThe corresponding postfix expression is :\n %s \n", postfix);

}

void evaluate()

{

            int i = 0, j = 0;

            char symb, temp;

            push('#');

            for(i=0; infix[i] != '\0'; i++)

            {

                        symb = infix[i];

                        switch(symb)

                        {

                                    case '(' :            push(symb);

                                                            break;

                                    case ')' :            temp = pop();

                                                            while(temp != '(' )

                                                            {

                                                                        postfix[j] = temp;

                                                                        j++;

                                                                        temp = pop();

                                                            }

                                                            break;

                                    case '+' :

                                    case '-' :

                                    case '\*' :

                                    case '/' :

                                    case '%' :

                                    case '^' :

                                    case '$'  :          while( prec(stack[top]) >= prec(symb) )

                                                            {

                                                                        temp = pop();

                                                                        postfix[j] = temp;

                                                                        j++;

                                                            }

                                                            push(symb);

                                                            break;

                                    default:            postfix[j] = symb;

                                                            j++;

                         }

            }

            while(top > 0)

            {

                        temp = pop();

                        postfix[j] = temp;

                        j++;

            }

            postfix[j] = '\0';

}

 void push(char item)

{

            top = top+1;

            stack[top] = item;

}

 char pop()

{

            char item;

            item = stack[top];

            top = top-1;

            return item;

}

int prec(char symb)

{

            int p;

            switch(symb)

            {

                        case '#' :           p = -1;

                                                break;

                        case '(' :

                        case ')' :            p = 0;

                                                break;

                        case '+' :

                        case '-' :            p = 1;

                                                break;

                        case '\*' :

                        case '/' :

                        case '%' :          p = 2;

                                                break;

                        case '^' :

                        case '$' :           p = 3;

                                                break;

            }

            return p;

}

***Output:***

Enter the valid infix expression:       **(a+b)+c/d\*e**

The entered infix expression is :

**(a+b)+c/d\*e**

The corresponding postfix expression is :

**ab+cd/e\*+**