```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include <string.h>
char infix_string[20], postfix_string[20];
int top; int stack[20]; int pop();
int precedence(char symbol);
int isEmpty();
void infix_to_postfix();
int check_space(char symbol);
void push(long int symbol);
int main()
{
   int count, length;
   char temp;
   top = -1;
   printf("\nINPUT THE INFIX EXPRESSION : ");
   scanf("%s", infix_string);
   infix_to_postfix();
   printf("\nEQUIVALENT POSTFIX EXPRESSION : %s\n", postfix_string);
   return 0;
}
void infix_to_postfix()
{
   int count, temp = 0;
   char next;
   char symbol;
   for(count = 0; count < strlen(infix_string); count++)
   {
      symbol = infix_string[count];
```

```
    if(!check_space(symbol))
    {
        switch(symbol)
        {
            case '(': push(symbol);
                break;
            case ')':
                while((next = pop()) != '(')
                {
                    postfix_string[temp++] = next;
                }
                break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '%':
            case '^':
                while(!isEmpty() && precedence(stack[top]) >= precedence(symbol))
                    postfix_string[temp++] = pop();
                push(symbol);
                break;
            default:
                postfix_string[temp++] = symbol;
        }
    }
}
while(!isEmpty())
{
    postfix_string[temp++] = pop();
```

```
    }
    postfix_string[temp] = '\0';
}
int precedence(char symbol)
{
    switch(symbol)
    {
        case '(': return 0;
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
        case '%':
            return 2;
        case '^':
            return 3;
        default:
            return 0;
    }
}
int check_space(char symbol)
{
    if(symbol == '\t' || symbol == ' ' )
    {
        return 1;
    }
    else
    {
        return 0;
```

```
    }
}
void push(long int symbol)
{
    if(top > 20)
    {
        printf("Stack Overflow\n");
        exit(1);
    }
    top = top + 1;
    stack[top] = symbol;
}
int isEmpty()
{
    if(top == -1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int pop()
{
    if(isEmpty())
    {
        printf("Stack is Empty\n");
        exit(1);
    }
    return(stack[top--]);
```

}

OUTPUT :


INPUT THE INFIX EXPRESSION : a+b*(c-d)/e
EQUIVALENT POSTFIX EXPRESSION : abcd-*e/+
...Program finished with exit code 0
Press ENTER to exit console.