Infix_to_Postfix\infix_to_postfix.c

```c
#include <stdio.h>
#define Maxsize 10
int top = -1;
char stack[50];

// Defining push function
void push(char ch)
{
    if (top >= Maxsize - 1)
    {
        printf("Stack is full \n");
    }
    else
    {
        top++;
        stack[top] = ch;
    }
}

//Defining pop function
char pop()
{
    if (top < 0)
    {
        printf("stack is empty.");
        return 'o';
    }
    else
    {
        char ch = stack[top];
        top--;
        return ch;
    }
}

//Assigning Priority to each character
int prority(char x)
{
    switch (x)
    {
    case '^':
        return 5;
        break;
    case '/':
    case '*':
        return 4;
        break;
    case '+':
```

```
    case '-':
        return 3;
        break;
    case '(':
        return 0;
        break;
    default:
        return -1;
        break;
    }
}

//Defining infix to postfix function
void infixtopostfix(char e[])
{
    char ch;
    for (int i = 0; e[i] != '\0'; i++)
    {
        if ((e[i] >= 'A' && e[i] <= 'Z') || (e[i] >= 'a' && e[i] <= 'z'))
        {
            printf("%c", e[i]);
        }
        else if (e[i] == '(')
        {
            push(e[i]);
        }
        else if (e[i] == ')')
        {
            ch = pop();
            while (ch != '(')
            {
                printf("%c", ch);
                ch = pop();
            }
        }
        else
        {

            while (prority(stack[top]) >= prority(e[i]))
            {

                printf("%c", ch);
                ch = pop();
            }
            push(e[i]);
        }
    }
    while (top >= 0)
    {
        ch = pop();
```

```c
        printf("%c", ch);
    }
}



int main()
{
    char e[Maxsize];
    printf("Enter a expression : ");
    scanf("%s", e);
    infixtopostfix(e);
    return 0;
}

```