2. WAP to convert a given valid parenthesized infix arithmatic expression to postfix expression. the expression consists of single character operand and binary operator (plus, minus, multiply & divide)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int Prec(char c)
{
 if(c=='^')
    return 3;
 else if(c=='/'||c=='*')
    return 2;
 else if(c=='+'||c=='-')
    return 1;
 else
    return -1;
}

char Associativity(char c)
{
 if(c=='^')
   return 'R';
 return 'L';
}

void infixToPostfix(const char*s)
{
 int len=strlen(s);
 char *result=(char*)malloc(len+1);
 char* stack=(char*)malloc(len);
 int resultIndex=0;
 int stackIndex=-1;

 if(!result||!stack)
 {
  printf("memory allocation failed!\n");
```

```
return;
}
for(int i=0;i<len;i++)
{
 char c=s[i];

 if((c>='a' && c<='z') || (c>='A' && c<='z') ||
  (c>='0' && c<='9'))
  {
    result[resultIndex++]=c;
  }
  else if(c=='(')
   {
     stack[++stackIndex]=c;
   }
  else if(c==')')
   {
    while(stackIndex>=0 && stack[stackIndex]
                          //remaining -> !='('
     {
      result[resultIndex++]=stack[stackIndex--];
     }
     stackIndex--;
   }
  else {
    while(stackIndex>=0 && (prec(c)<prec(stack[
          stackIndex]) || (prec(c)==prec(stack[stack
          Index])==='L')))
    {
      result[resultIndex++]=stack[stackIndex--];
    }
  stack[++stackIndex]=c;
  }
}
```

```
while(stackIndex >=0)
 {
  result[resultIndex++]= stack[stackIndex--];
 }
 result[resultIndex] ='\0';
 printf("%s\n", result);
 free(result);
 free(stack);
  }
 int main()
  {
  char exp[] = "a+b*(c^d-e)^(f+g*h)-i";
  infixTOpostfix(exp);
  return 0;
 }
```

```
scanf
char exp[];
printf("enter expression
scanf("%s", &exp);
```

Seen.

O/P

abcd^e-fgh*+^*i-

Explanation

Postfix syntax: <operand> <operand> <op

a+b*(c^d-e)^(f+g*h)-i

= a

= a+b*(

O/P = a+b(

<operand> <operand> <operator>

<operator> <operand> <operand>

<operand> <operator> <operand>

<op

a+b*(c^d-e)^(f+g*h)-i

* a+b*cd^e-^fgh*+^-i

* cd^e-fgh*+^

* a+bcd*cd^e^efgh*+^-i

* a+bcd^efgh*+^*-i

* abcd^efgh*+^+-i

* abcd^efgh*+^*+i-;

This is infix to postfix equation.

O/P

The Expression of infix to postfix is : abcd^e-fgh*+^*i-

b

Method 2

a+b*(c^d-e)^(f+g*h)-i

	Stack	Expression
a		a
+	+	a
b		ab
(	+(	ab
*	+(*	ab
c	+(*	abc
^	+(*^	abc
d	+(*^	abcd
-	+(*^-	abcd^
e	+(*-	abcd^e
)	+(*-)	abcd^e
^	+(*-)^	abcd^e
(	+(*-)^(	abcd^e
f	+(*-)^(	abcd^ef
+	+(*-)^(+	abcd^ef
g	+(*-)^(+	abcd^efg
*	+(*-)^(+	abcd^efgh*+
h	+(*^	abcd^efgh*+^
-	+*^-	abcd^efgh*+^
i	*^-	abcd^efgh*+^i
	-	abcd^efgh+^*i-