Program 2

Write a program to convert a given valid parenthesized in-fix arithmetic expression to post-fix expression. The expression consists of single character operands and the binary operators +,-,*,/.

Observation:

```
to convert given valled palanthesized infix authmatic
expression to postfex expression.

=> ## include < s+dio.h>

## include < s+dlab.h>
   # include < string.h)
   ent prec (charc) {
      ef (c=z'^')
        return 3;
      ebeif (c== 1 11. c=z*)
      return 2;
else if ( c=='f' || c=='-')
       return 1:
      else
       return -1; }
     char associativity (charc) {

if (c== 1)

neturn 'R';
         return i; Il Default to left associative
    void infixtopositix (const chas *s){
         int len 2 stalen (s);
       char* result = (char*) malloc (lenti);
       Char & stack = (char*) malloc (len);
         int result Index = 0;
       int stack Index = 0; -1;
         if (! result 11 ! stack){

[printf ("Memory allocation failed)n");
              return;
```

```
Char Cz scij;
       if (c>='a'ff c <='z') || (c>='A'ff ec='z') ||
            (c>='0' ff c <='9'))
           resut [restIndex++] = C;
       else if (C==')'){
         while (stack Index > 20 ffstack [stack Index] = '()
         result [sesult Industi] = stack [stack Indus-];
         stackIndux --;
      elsef
       while (StackIndex) 20 ff (PRECCE) < pre (Stack Istacksnow
               11 (psec (c) = = psec (stack [stack Indix + f]) ff
               associativity (c) = 2'L')))
     result [sesult Index +] = stack[stackIndex -]
           stack[t+stackIndex]=C;
   while (stack Index >=0) {
       result [sesultIndex++] = stack[stackIndex-]
result [resultIndex]='10';
       printf ("%s|n", result);
```

free (result); free (stack); } every to notation of queue {; (shark) int main () { charexp[] = "a+b*(c^d-e)^(f+g*h)-it; infixTo postfix (exp); return 0; Ef (Is Full ()) Penty ("Queve is full"); Output abcdre-fgh*+1*+1- (1) ptpm3 21) je oelo front=0. read=0. Seen Pequeue () It (Is Empty (1) Mitty (" Que ?s trupty"); else ef (tront == reas) front - room - si front = front 1 ;

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int prec(char c){
    if (c == '+' || c == '-') {
        return 1;
    if (c == '*' || c == '/') {
    if (c == '^') {
        return 3;
    return 0;
char associativity(char c){
    if(c=='^')
        return 'R';
    return 'L';
void infixtopostfix(const char *s){
    int len = strlen(s);
    char result[len+1];
    char stack[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'))
            result[resultIndex++] = c;
        else if(c=='('){
            stack[++StackIndex]=c;
        else if(c==')'){
           while (StackIndex>=0 && stack[StackIndex]!='(')
```

```
{
    result[resultIndex++]=stack[StackIndex--];
    }
    StackIndex--;
}
else{
    while (StackIndex>=0 && (prec(c))<prec(stack[StackIndex]) ||
(prec(c)==prec(stack[StackIndex]) && associativity(c)=='L'))
    {
        result[resultIndex++]=stack[StackIndex--];
    }
    stack[++StackIndex]=c;
}
while (StackIndex>=0)
{
    result[resultIndex++]=stack[StackIndex--];
}
result[resultIndex++]='\0';
printf("%s\n",result);
}
int main(){
    char exp[] = "a+b*(c^d-e)^(f+g*h)-i";
    infixtopostfix(exp);
    return 0;
}
```

Output:

```
PS C:\Users\satis> & 'c:\Users\satis\.vscode\extensions\
tdin=Microsoft-MIEngine-In-b44lhpi0.gsx' '--stdout=Micros
d=Microsoft-MIEngine-Pid-txx4ou1c.bvi' '--dbgExe=C:\msys6
abcd^e-fgh*+^*+i-
PS C:\Users\satis>
```