PROGRAM 4

**Design, develop and implement 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.**

<u>Program objective:</u>

- Understand different notations to represent regular expression.
- Understand infix to postfix conversion.
- Understand the precedence of operators.

Algorithm:

Step 1: Read the infix expression as a string.

Step 2: Scan the expression character by character till the end. Repeat the following operations

1. If it is an operand add it to the postfixexpression.
2. If it is a left parenthesis push it onto the stack.
3. If it is a right parentheses pop out elements from the stack and assign it to the postfix string. Pop out the left parentheses but don’t assign topostfix.

Step 3: If it is an operator compare its precedence with that of the element at the top of stack.

1. If it is greater push it onto the stack.
2. Else pop and assignelements in the stack to the postfixexpression until you find one such element.

Step 4: If you have reached the end of the expression, pop out any leftover elements in the stack till it becomes empty.

Step 5: Append a null terminator at the end display the result

THEORY

Infix: Operators are written in-between their operands. Ex: X + Y

Prefix: Operators are written before their operands. Ex: +X Y **postfix:** Operators are written after their operands. Ex: XY+

Examples of Infix, Prefix, and Postfix

Infix Expression	Prefix Expression	Postfix Expression
A + B	+ A B	A B +
A + B * C	+ A * B C	ABC*+

Infix to prefix conversion Expression = **(A+B^C)*D+E^5**

Step 1. Reverse the infix expression.

5^E+D*)C^B+A(

Step 2. Make Every '(' as ')' and every ')' as '('

5^E+D*(C^B+A)

Step 3. Convert expression to postfix form.

Step 4. Reverse the expression.

+*+A^BCD^E

Step 5. Result

+*+A^BCD^E5

PROGRAM:

```
#include<stdio.h>
#include<ctype.h>
#define SIZE 50
char s[SIZE];
int top=-1;
 void push(char elem)
 {
  s[++top]=elem;
 }
char pop()
{
return s[top--];
}
int pr(char elem)
{
switch(elem)
{
case '#':return 0;
case '(':return 1;
case '+':
case '-':return 2;
case '*':
case '/':
case '%':return 3;
case '^':return 4;
}
}
void main()
{
char infix[50],postfix[50],ch,elem;
int i=0,k=0;
printf("enter the  infix expression\n");
gets(infix);
push('#');
while((ch=infix[i++])!='\0')
```

```
{
if(ch=='(')
push(ch);
else if(isalnum(ch))
postfix[k++]=ch;
else if(ch==')')
{
while(s[top]!='(')
postfix[k++]=pop();
elem=pop();
}
else
{
while(pr(s[top])>=pr(ch))
postfix[k++]=pop();
push(ch);
}
}
while(s[top]!='#')
postfix[k++]=pop();
postfix[k]='\0';
printf("infix expression is %s\n postfix expression is %s\n",infix,postfix);
}
```

Output1

```
enter the Infix Expression
((a+b)*c)
Given Infix Expn is: ((a+b)*c)
The Postfix Expn is: ab+c*
```

Output 2

```
enter the Infix Expression
(a+ (b-c)*d)
Given Infix Expn is: (a+ (b-c)*d)
The Postfix Expn is: abc-d*+
```

<u>Program outcome :</u>

- Identify the applications of infix and postfix.
- Implement C program to convert infix to postfix.
- Identify the different operators.

<u>Viva Questions:</u>

- What is a postfix expression?
- What are Infix, prefix, Postfix notations?
- What is the evaluation order according to which an infixexpressionis converted to postfix expression ?
- which data structure is used for infix to postfix conversion

PROGRAM 5

Design, develop and implement a Program inC for the following StackApplications

a. Evaluation of Suffixexpression withsingle digit operandsand operators: +, -, *, /,%, ^

b. Solving Tower of Hanoi problem with n disks

<u>Program objective :</u>

- Understand different polish notation.
- Understand the methodology of evaluating suffix expression.
- Get the knowledge of operator precedence and associativity.

Algorithm

```
Step 1: Read the suffix/postfix expression
Step 2: Scan the postfix expression from left to right character by character
Step 3: if scanned symbol is operand push data into stack.
        If scanned symbol is operator pop two elements from stack Evaluate result
        and result is pushed onto stack
Step 4: Repeat step 2-3 until all symbols are scanned completely
```

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#define MAX 50
char post[MAX];
int stack[MAX],top=-1,i;
void pushstack(int);
void calculator(char);
main()
{
 printf("enter suffix expression\n");
 gets(post);
 for(i=0; i<strlen(post); i++)
 {
  if(post[i]>'0'&& post[i]<='9')
  pushstack(i);
  else
   calculator(post[i]);
  }
   printf("result=%d\n",stack[top]);
 }
void pushstack(int i)
 {
   top=top+1;
   stack[top]=(int)(post[i]-48);
 }
void calculator(char c)
 {
  int a,b,ans;
  b=stack[top--];
  a=stack[top--];
  switch(c)
  {
   case '+':ans=a+b;break;
   case '-':ans=a-b;break;
   case '*':ans=a*b;break;
   case '/':ans=a/b;break;
   case '%':ans=a%b;break;
   case '^':ans=pow(a,b);break;
   default :printf("wrong input\n");
   exit(0);
  }
   top++;
   stack[top]=ans;
 }
```

Output1

```
enter suffix expression:
23+
The result is 5
```

Output2

```
enter suffix expression:
123-4*+
The result is -3.
```

Output3

```
enter suffix expression:
623+-382/+*2$3+
The result is 52
```

<u>Program outcome:</u>

- Identify the applications of suffix expression.
- Familiarized with the methodology of suffix evaluation.
- Familiarized the operator precedence and associativity.

<u>Viva Questions</u>

- What is Suffix Expression?