PROGRAM 2

Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide) -

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 50

char stack[SIZE];
int top = –1;

void push(char elem)
{
    int val;

    if (top == SIZE – 1)
    {
        printf("\nStack overflow!!");
    }
    else
    {
        stack[++top] = elem;
    }
}

char pop()
```

```
{
    if (top == -1)
    {
        printf("\nStack underflow!!");
    }
    else
    {
        int popele = stack[top--];
        return popele;
    }
}

int pr(char symbol)
{
    if (symbol == '^')
    {
        return (3);
    }
    else if (symbol == '*' || symbol == '/')
    {
        return (2);
    }
    else if (symbol == '+' || symbol == '-')
    {
        return (1);
    }
    else
    {
        return (0);
```

```
    }
}

int main()
{
    char infix[50], postfix[50], ch, elem;
    int i = 0, k = 0;

    printf("Enter the Infix expression: ");
    scanf("%s", &infix);

    push('#');

    while ((ch = infix[i]) != '\0')
    {
        if (ch == '(')
            push(ch);
        else if (isalnum(ch))
            postfix[k++] = ch;
        else if (ch == ')')
        {
            while (stack[top] != '(')
                postfix[k++] = pop();
            elem = pop();
        }
        else
        {
            while (pr(stack[top]) >= pr(ch))
                postfix[k++] = pop();
```

```
            push(ch);
        }
        i++;
    }
    while (stack[top] != '#')
        postfix[k++] = pop();

    postfix[k] = '\0';
    printf("\nPostfix Expression = %s\n", postfix);

    return 0;
}
```

Output –