Program 2 : Design 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.

```
#define SIZE 50 /* Size of Stack */

#include <ctype.h>
#include <stdio.h>

char s[SIZE];
int top = -1; /* Global declarations */

void push(char elem) /* Function for PUSH operation */
{
        s[++top] = elem;
}

char pop() /* Function for POP operation */
{
        return (s[top--]);
}

int pr(char elem) /* Function for precedence */
{
        switch (elem)
        {
```

```
                case '#':       return 0;
                case '(':       return 1;
                case '+':
                case '-':       return 2;
                case '*':
                case '/':
                case '%':       return 3;
                case '^':       return 4;
        }
}

void main() /* Main Program */
{
        char infx[50], pofx[50], ch, elem;
        int i = 0, k = 0;

        printf("\n\nRead the Infix Expression ? ");
        scanf("%s", infx);

        push('#');

        while ((ch = infx[i++]) != '\0')
        {
                if (ch == '(')
                    push(ch);
                else if (isalnum(ch))    /* Check character Alpha Numeric  */
                    pofx[k++] = ch;
```

```
        else if (ch == ')')
        {
                while (s[top] != '(')
                    pofx[k++] = pop();
                elem = pop(); /* Remove ( */
         }
        else /* Operator */
        {
                while (pr(s[top]) >= pr(ch))
                     pofx[k++] = pop();
                 push(ch);
         }
    }

    while (s[top] != '#')          /* Pop from stack till empty */
    pofx[k++] = pop();
    pofx[k] = '\0';                /* Make pofx as valid string */
          printf("\n\nGiven Infix Expn: %s \n Postfix Expn: %s\n", infx,
     pofx);
    getch();
}
```