

12. Write a C program to implement the application of Stack (Notations)

```
#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

void push(char x) {
    if (top == SIZE - 1)
        printf("Stack Overflow\n");
    else
        stack[++top] = x;
}

char pop() {
    if (top == -1)
        return -1;
    else
        return stack[top--];
}

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

```

void infixToPostfix(char infix[], char postfix[]) {
    char *e, x;
    e = infix;
    int k = 0;
    while (*e != '\0') {
        if (isalnum(*e)) {
            postfix[k++] = *e;
        } else if (*e == '(') {
            push(*e);
        } else if (*e == ')') {
            while ((x = pop()) != '(')
                postfix[k++] = x;
        } else {
            while (priority(stack[top]) >= priority(*e))
                postfix[k++] = pop();
            push(*e);
        }
        e++;
    }
    while (top != -1)
        postfix[k++] = pop();

    postfix[k] = '\0';
}

int evalPostfix(char postfix[]) {
    int evalStack[SIZE], evalTop = -1;
    char *e;
    int n1, n2, n3;
    e = postfix;
    while (*e != '\0') {
        if (isdigit(*e)) {

```

```

        evalStack[++evalTop] = *e - '0';
    } else {
        n1 = evalStack[evalTop--];
        n2 = evalStack[evalTop--];
        switch (*e) {
            case '+':
                n3 = n2 + n1;
                break;
            case '-':
                n3 = n2 - n1;
                break;
            case '*':
                n3 = n2 * n1;
                break;
            case '/':
                n3 = n2 / n1;
                break;
        }
        evalStack[++evalTop] = n3;
    }
    e++;
}

return evalStack[evalTop];
}

int main() {
    char infix[SIZE], postfix[SIZE];
    printf("Enter Infix Expression: ");
    scanf("%s", infix);
    infixToPostfix(infix, postfix);
    printf("Postfix Expression: %s\n", postfix);
}

```

```

printf("Result after Evaluation: %d\n", evalPostfix(postfix));

return 0;

}

```

main.c	Output
<pre> 1 #include &lt;stdio.h&gt; 2 #include &lt;ctype.h&gt; 3 #include &lt;string.h&gt; 4 #include &lt;stdlib.h&gt; 5 #define SIZE 100 6 char stack[SIZE]; 7 int top = -1; 8 void push(char x) { 9     if (top == SIZE - 1) 10         printf("Stack Overflow\n"); 11     else 12         stack[++top] = x; 13 } 14 char pop() { 15     if (top == -1) 16         return -1; 17     else 18         return stack[top--]; 19 } 20 int priority(char x) { 21     if (x == '(') 22         return 0; 23     if (x == '+'    x == '-') 24         return 1; 25     if (x == '*'    x == '/') 26         return 2; 27     if (x == '^') 28         return 3; </pre>	<pre> Enter Infix Expression: (3+5)*9 Postfix Expression: 35+9* Result after Evaluation: 72  === Code Execution Successful === </pre>