12. Application of Stack

Aim:

To write a C program to convert infix expression to postfix using a stack.

Algorithm:

1. Start program.
2. Scan infix expression from left to right.
3. If operand → add to postfix.
4. If operator → push to stack (precedence rules).
5. If ')' → pop till '('.
6. End with postfix expression.

Code:

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 100

char stack[SIZE];
int top = -1;

void push(char c) { stack[++top] = c; }
char pop() { return stack[top--]; }
int precedence(char c) {
    if (c == '^') return 3;
    if (c == '*' || c == '/') return 2;
    if (c == '+' || c == '-') return 1;
    return -1;
}

void infixToPostfix(char* exp) {
    char postfix[SIZE];
```

```
    int i = 0, k = 0;
    char c;
    while ((c = exp[i++]) != '\0') {
        if (isalnum(c))
            postfix[k++] = c;
        else if (c == '(')
            push(c);
        else if (c == ')') {
            while (top != -1 && stack[top] != '(')
                postfix[k++] = pop();
            pop();
        } else {
            while (top != -1 && precedence(stack[top]) >= precedence(c))
                postfix[k++] = pop();
            push(c);
        }
    }
    while (top != -1)
        postfix[k++] = pop();
    postfix[k] = '\0';
    printf("Postfix: %s\n", postfix);
}

int main() {
    char exp[] = "A+B*C";
    printf("Infix: %s\n", exp);
    infixToPostfix(exp);
    return 0;
}
```

Sample Output:

```
Infix: A+B*C
Postfix: ABC*+


=== Code Execution Successful ===
```

Result:

Successfully converted infix to postfix using stack.