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

**Aim**

To write a C program that uses a stack to convert an **infix** expression (with operators between operands) into its **postfix** form (also called Reverse Polish Notation), and optionally prefix form. This demonstrates the use of stack data structure for expression notation conversion.

**Algorithm**

We’ll consider converting **infix → postfix**. The algorithm uses a stack to temporarily hold operators and manage precedence and parentheses.

1. Initialize an empty stack (for operators) and an output string (postfix).
2. Scan the infix expression from left to right, character by character.
3. For each symbol ch in the infix expression:
   * If ch is an operand (e.g. letter or digit), append it to postfix output.
   * Else if ch is an opening parenthesis '(', push it onto the stack.
   * Else if ch is a closing parenthesis ')', then:
     + Pop operators from stack and append to output until an opening parenthesis '(' is at the top.
     + Pop and discard the '('.
   * Else if ch is an operator (+, -, \*, /, ^, etc.), then:
     + While stack is not empty and the precedence of operator at stack top is **greater than or equal** to the precedence of ch (also taking associativity into account), pop from stack and append to output.
     + Push ch onto the stack.
4. After the input is fully scanned, pop any remaining operators from the stack and append to output.
5. Output the postfix expression.

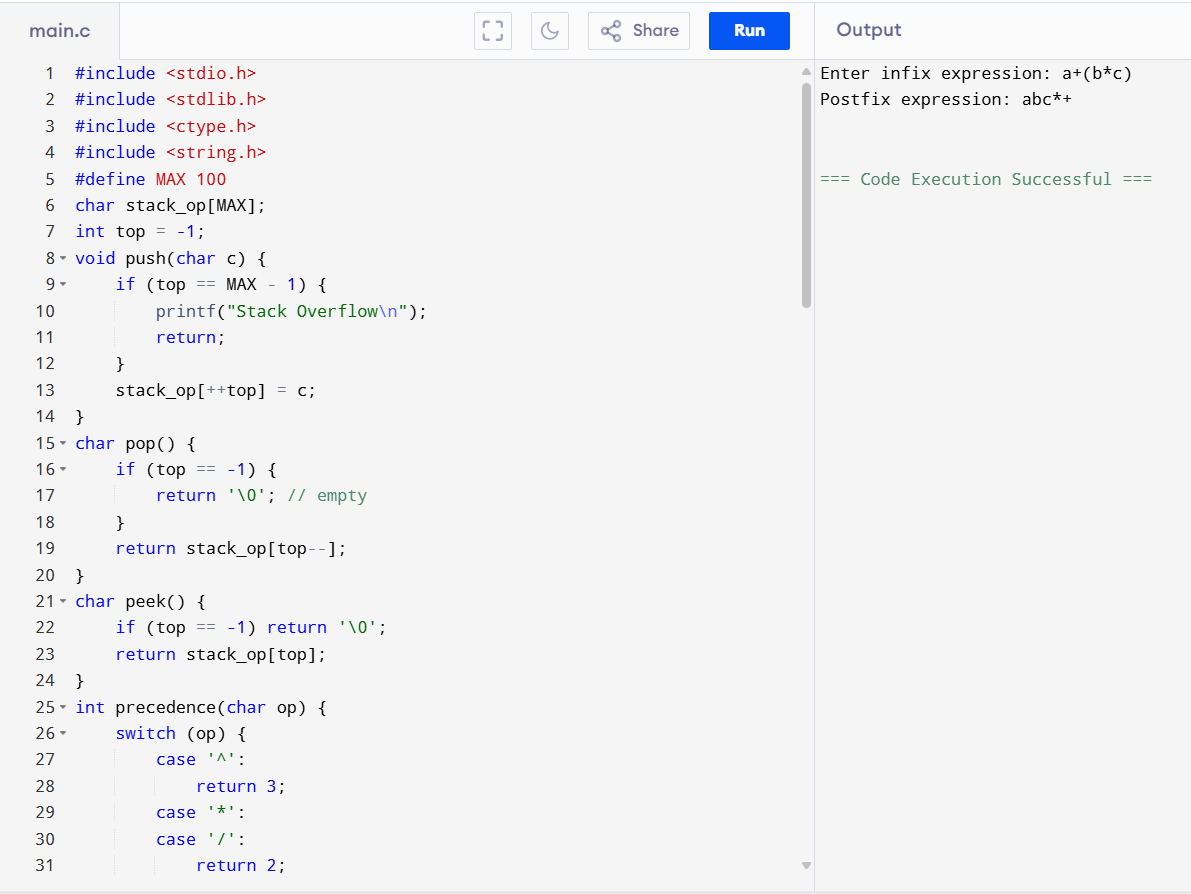
**Precedence rules** (common):

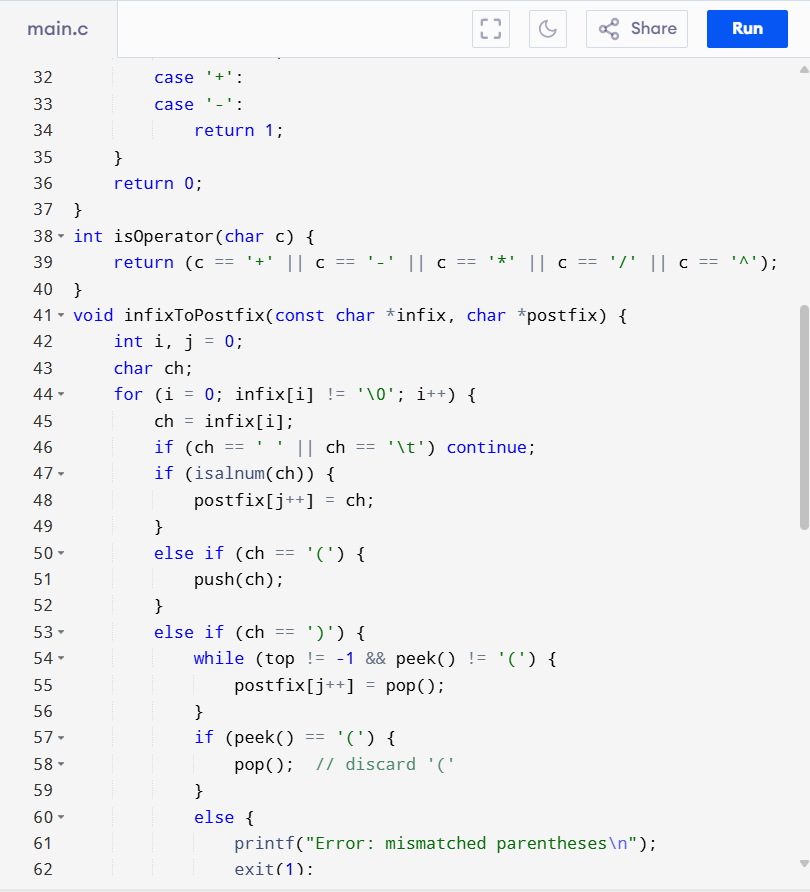
* ^ (exponentiation) – highest (often **right-associative**).
* \*, / – next.
* +, - – lowest among explicit binary operators.
* Parentheses override.

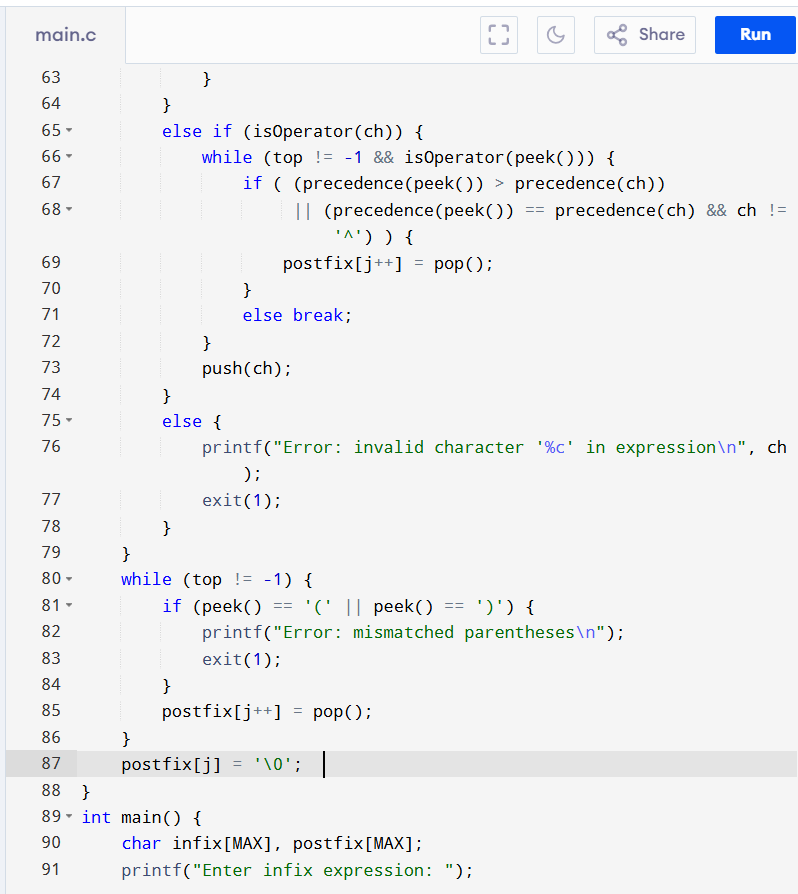
**Associativity**:

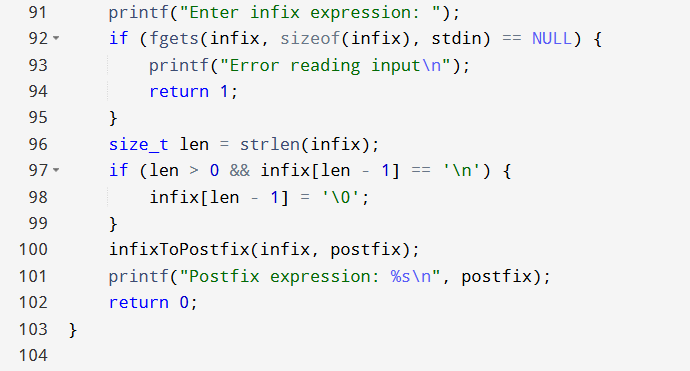
* ^ often right‐associative: so if new operator is ^ and top of stack is also ^, you might treat differently.
* \*, /, +, - are left associative: so while top of stack has *equal or greater* precedence, pop.

**Program**

****

****

****

****

**Result:** The program has been executed successfully.