1. Write a program to implement a stack of integers using a **fixed-size array**. The program should support the following operations:
    - `push(int)` – Add an element
    - `pop()` – Remove and return the top element
    - `peek()` – View the top element without removing
    - `isEmpty()` – Return true if the stack is empty
    - `isFull()` – Return true if the stack is ful

2. Write a program to reverse a given array of integers using a stack implemented with arrays. Do not use built-in reverse or any extra array logic — only stack operations.

3. Write a program to check whether a given string of parentheses is balanced or not. The string may contain `()`, `{}`, and `[]`. Use a stack to solve this.

4. Write a program to convert an infix expression (e.g., `a + b * (c - d)`) to postfix notation using a stack.

5. Write a program to evaluate a postfix expression like `5 3 + 2 *`. Use a stack to store operands and compute the result.

6. Simulate browser navigation using **two stacks**:

    - One for the **back history**
    - One for the **forward history**

   Your program should allow:

    - `visit(URL)`
    - `back()`
    - `forward()`
    - `showCurrent()`

7. Create a **text editor simulation** where each typing action is stored in a stack. Allow:

- `type("text")`
- `undo()` – restores the previous state
- `redo()` – re-applies undone action

Use:

- One stack for undo history
- One stack for redo history