### **Task 1: Evaluate Postfix Expression**

**Objective**: Write a program to evaluate a postfix (Reverse Polish Notation) mathematical expression using a stack.

**Steps**:

1. Read a postfix expression as input (e.g., "23+5\*", which means (2 + 3) \* 5).
2. Use a stack to evaluate the expression:

* Push numbers onto the stack.
* When encountering an operator, pop two numbers, apply the operator, and push the result back onto the stack.

1. At the end, the stack should contain a single number, which is the result.

**Example Input/Output**:

* Input: 23+5\*
* Output: 25

### **Task 2: Implement a Browser Back/Forward Navigation**

**Objective**: Simulate a simple browser navigation system using two stacks.

**Steps**:

1. Create two stacks: back\_stack and forward\_stack.
2. Implement the following operations:

* visit(url) - Navigate to a new URL and clear the forward\_stack.
* back() - Move to the previous URL (pop from back\_stack and push to forward\_stack).
* forward() - Move to the next URL (pop from forward\_stack and push to back\_stack).

1. Print the current URL after each operation.

**Example Input/Output**:

visit("google.com")

visit("youtube.com")

back() # Output: google.com

forward() # Output: youtube.com

### **Task 3: Check for Palindrome Using Stack**

**Objective**: Write a program to check if a string is a palindrome using a stack.

**Steps**:

1. Push all characters of the string onto a stack.
2. Pop characters from the stack to create a reversed string.
3. Compare the reversed string with the original string to determine if it is a palindrome.

**Example Input/Output**:

* Input: "madam"
* Output: True
* Input: "hello"
* Output: False

**Best of Luck ☺**