Data structure Exercise 3 for stack

1. Stack:
   - Q1.Challenges: push [“1”,”2”,”3”,”4”], pop3. Which remains?

     Instructions: *For the challenge questions, design the solution in a clear algorithmic sequence*

     and explain each step alongside the corresponding code lines.

     Answer: alogarithmic seqwence step by step

     Step 1: Understanding the Problem

     We have a **stack**, which works like a **pile of books**: the last book you put on top is the first one you remove (LIFO – Last In, First Out).

     We will:

1. Push "1", "2", "3", "4" into the stack.
2. Remove (pop) the top 3 items.
3. See what is left.

Step 2: How it Works Step by Step

- **Start with an empty stack: [].**
- **Push elements one by one:**
  - **Push "1" → Stack: ["1"]**
  - **Push "2" → Stack: ["1", "2"]**
  - **Push "3" → Stack: ["1", "2", "3"]**
  - **Push "4" → Stack: ["1", "2", "3", "4"]**
- **Pop 3 times (remove from top):**
  - **Pop 1 → remove "4" → Stack: ["1", "2", "3"]**
  - **Pop 2 → remove "3" → Stack: ["1", "2"]**
  - **Pop 3 → remove "2" → Stack: ["1"]**
- **Remaining element = "1"**

Code step by step

- stack = []

- for item in ["1", "2", "3", "4"]:
- stack.append(“1”)
- stack.append(“2”)
- stack.append(“3”)
- stack.append(“4”)
- for _ in range(3):
- stack.pop()
- stack.pop()
- stack.pop()
- print("Remaining stack:", stack)

Q2. • Reflection: Why stack can’t manage queues of customers?
Answer:

A stack works like a pile of plates where the last plate you put on top is the first one you take off. This is called LIFO (Last In, First Out). A queue, however, is like people standing in line: the first person to arrive is the first to be served (FIFO). If we used a stack for a queue, the last person to arrive would be served before everyone else, which is unfair. That’s why stacks cannot manage real queues of people—they reverse the order, while queues need to keep it.