3.Challenge 1 (Stack)

Task: Reverse “URCAMPUS” using a stack

Algorithm

1. Input string s = “URCAMPUS”.
2. Initialize empty stack.
3. Iterate over each character of s and push onto stack.
4. Initialize empty list reversed_chars.
5. While stack not empty:
   - Pop character from top.
   - Append to reversed_chars.
6. Concatenate characters in reversed_chars to create final reversed string.
7. Display both original and reversed strings.

In code:

```
# Step 1: input
s = "URCAMPUS"                          # (1) original string

# Step 2: initialize empty stack
stack = []                              # (2)

# Step 3: push each character
for ch in s:                            # (3)
    stack.append(ch)                    # (3) push operation

# Step 4: pop to reverse
reversed_chars = []                     # (4)
while stack:                            # (5)
    reversed_chars.append(stack.pop())  # (5) pop operation

# Step 6 & 7: join + output
reversed_s = "".join(reversed_chars) # (6)
print("Original:", s)                # (7)
print("Reversed:", reversed_s)       # (7)
```

4.Reflection

• A stack follows the LIFO principle — the last character pushed is the first one popped.

• That property automatically reverses the order of characters when we pop them out.

• Real-life connection: In Rwanda, imagine stacking plates in a canteen — the last plate to be placed on top is the first one to be taken by the next student. Similarly, "URCAMPUS" spelled backwards is "SUPMACRU".

The Stacks are perfect for problems or situations requiring reversal or undo operations.