Names: SHEJA KEVIN Alberto

Reg №: 224004070

Class: BIT Level 2

Date: 24th September 2025

Subject: DATA STRUCTURE AND ALGORITHM

N.B : *Screenshots of the following codes are in the screenshot folder.*

PROJECT 102 – STACKS AND QUEUES

Stack Questions

1. Practical (Rwanda): Irembo Example

```
stack = []
stack.append("Upload ID")
stack.append("Fill Form")
stack.append("Confirm")
stack.pop()
print(stack)
```

Output:

['Upload ID', 'Fill Form']

After undoing one action, only ***Upload ID*** and ***Fill Form*** remain.

Explanation:

When using Irembo, you normally go through steps like ***uploading your ID*****,** then ***filling in the form*****,** and finally ***confirming the details.*** If we treat these steps like a stack, they are placed in order as: **Upload ID, Fill Form, Confirm**.

Now, if you undo one step, the last thing you did, which is "**Confirm,**" is the one that gets removed first. After undoing, you're left with **Upload ID and Fill Form**.

This is a good example of how stacks can be applied in real life.

2. Practical (Rwanda): UR Example

```
stack = []
stack.append("TopicA")
stack.append("TopicB")
stack.append("TopicC")
stack.pop()
stack.pop()
stack.pop()
print(stack)
```

Output:

```
[]
```

After popping all, the stack is **empty**.

Explanation:

At the University of Rwanda, think of a student revising by stacking topics: **TopicA, TopicB,** and **TopicC.** The last topic added (**TopicC**) is at the top.

If the student pops everything, the topics are removed one by one, starting with TopicC, then TopicB, and finally TopicA**.**

At the end, the stack is **empty**. This shows how stacks clear out in the exact reverse order that items were added.

3. Challenge

Algorithmic Steps:

1. Push A, B, C, D into the stack: Top = D
2. Pop two elements: Removes D and C
3. Push E: Stack becomes [A, B, E]
4. The top element is E

Python Code:

```
stack = ["A", "B", "C", "D"]

stack.pop()  # removes D

stack.pop()  # removes C

stack.append("E")

print("Top element:", stack[-1])
```

Output:

Top element: E

Explanation:

Imagine you push the letters **A, B, C,** and **D** into a stack. D is at the top. If you pop two items, C and D are removed, leaving A and B. After that, if you push E, the stack becomes **A, B, E** with E now sitting on top.

So, the **top element is E**.

4. Reflection

Stacks are perfect for undo operations because they follow **Last In, First Out**. Whatever you did last is always the first thing to be undone. For example, in Microsoft Word, if you type a sentence and then highlight it, pressing undo will remove the highlight first, not the whole sentence.

That's why stacks fit naturally into undo features.

QUEUE QUESTIONS

1. Practical (Rwanda): Airtel Example

```
from collections import deque

queue = deque(["Client1", "Client2", "Client3", "Client4", "Client5"])

queue.popleft()

print("Next client:", queue[0])
```

Output:

Next client: Client2

After serving one, Client2 is next.

Explanation:

At an Airtel service center, clients line up for help. If there are five people in the queue, the one at the very front is served first. After that, the second person automatically becomes next.

This shows how queues work in a **first-come, first-served** way.

2. Practical (Rwanda): CHUK Example

```
from collections import deque

queue =
deque(["Patient1","Patient2","Patient3","Patient4","Patient5","Patient6","Patient7
","Patient8","Patient9"])

queue.popleft()

queue.popleft()

print("Third patient served:",queue[0])
```

Output:

Third patient served: Patient3

Explanation:

At CHUK hospital, If nine patients are waiting in line. The doctor serves them in the order they arrived: first patient one, then patient two, then patient three.

So, the **third patient in the line is the third one to be served**.

3. Challenge

Algorithmic Steps:

1. People arrive for tickets.
2. Queue stores them in order of arrival.
3. Serving follows FIFO: first person in → first person out.
4. Using a stack would reverse the order, making it unfair.

Python Code:

```
from collections import deque

queue = deque(["Person1","Person2","Person3"])

print("Tickets go to:", list(queue))
```

Output:

Tickets go to: ['Person1', 'Person2', 'Person3']

Explanation:

If people are buying tickets for an event, the only fair system is a queue. Whoever came first gets served first. Using a stack would mean the last person to arrive gets the first ticket, which would cause arguments and confusion.

4. Reflection

Using FIFO in queues avoids chaos because it gives everyone confidence that they'll be served in the order they arrived. At big events like concerts, this order keeps things peaceful since no one can skip the line.

Queues make the process fair and organized, while keeping people calm.