Name: HABUMUGISHA SHEMA Jules Pascal

Registration number: 224005755

Class Number: 23

ASSIGNMENT OF DATA STRUCTURE PROJECT 23

STACK

1. Practical Question 1 (MoMo Example)

Problem:

Push ["Dial", "Enter PIN", "Confirm"]. Undo twice. Which step remains?

Code:

```
# Practical 1: MoMo Example (Stack)
stack = []  # empty stack

# Push steps
stack.append("Dial")
stack.append("Enter PIN")
stack.append("Confirm")

# Undo twice (Pop twice)
stack.pop()  # removes "Confirm"
stack.pop()  # removes "Enter PIN"

# Remaining step
print("Remaining step on stack:", stack)
```

Output (Screenshot Style):

```
Remaining step on stack: ['Dial']
```

✅ **Answer:** Only **“Dial”** remains.

---

2. Practical Question 2 (UR Student Example)

Problem:

Push ["Assignment", "Revision", "Group Work"]. Pop two. Which is left?

Code:

```
# Practical 2: UR Student Example (Stack)
stack = []  # empty stack

# Push tasks
stack.append("Assignment")
stack.append("Revision")
stack.append("Group Work")

# Pop twice
stack.pop()  # removes "Group Work"
stack.pop()  # removes "Revision"

# Remaining task
print("Remaining task on stack:", stack)
```

Output (Screenshot Style):

```
Remaining task on stack: ['Assignment']
```

✅ **Answer:** Only **“Assignment”** remains.

---

3. Challenge Question

Problem:

Push ["1","2","3","4"], pop twice, push "5". Show the **top** after operations.

Algorithm Steps:

1. Start with an empty stack.
2. Push 1, 2, 3, and 4 (stack now [1,2,3,4]).
3. Pop twice (remove “4” and “3”). Stack now [1,2].
4. Push “5” (stack now [1,2,5]).
5. Show top (last element in the list).

```
# Challenge Question: Stack operations
stack = []                          # Step 1: Empty stack

# Step 2: Push "1","2","3","4"
stack.append("1")
stack.append("2")
stack.append("3")
stack.append("4")

# Step 3: Pop twice
stack.pop()  # removes "4"
stack.pop()  # removes "3"

# Step 4: Push "5"
stack.append("5")

# Step 5: Show top element
print("Top of the stack is:", stack[-1])
```

Output (Screenshot Style):

```
Top of the stack is: 5
```

✅ **Answer:** The top of the stack is **“5”**.

4. Reflection Question

Question: Why does stack reflect “last action undone first”?

Answer (Theory Only):

A stack works on the **Last In, First Out (LIFO)** principle.

- The last item pushed onto the stack sits on top.
- When an undo or pop operation is performed, the **most recent action** (the one on top) is removed first.
  This mirrors how many real-world actions work — for example, in mobile apps or text editors, the **last action** you did is the **first one** to be undone when you press **Undo**. This behavior ensures actions can be reversed step by step in reverse order of how they were done.

QUEUE

1. Practical Question 1 (RRA Example)

Problem:
At RRA, 6 people join. After 1 is served, who is next?

Code:

```
# Practical 1: RRA Queue
from collections import deque

# Create an empty queue
queue = deque()

# 6 people join (P1–P6)
queue.append("P1")
queue.append("P2")
queue.append("P3")
queue.append("P4")
queue.append("P5")
queue.append("P6")

# Serve one person (pop from left)
queue.popleft()  # removes P1

# Who is next in line?
print("Next person to be served:", queue[0])
```

Output (Screenshot Style):

```
Next person to be served: P2
```

✅ **Answer: P2** is next.

---

2. Practical Question 2 (BK ATM Example)
Problem:
At BK ATM, 4 customers enqueue. After 2 are served, who is in front?

Code:

```python
# Practical 2: BK ATM Queue
from collections import deque

# Create an empty queue
queue = deque()

# 4 customers join (C1–C4)
queue.append("C1")
queue.append("C2")
queue.append("C3")
queue.append("C4")

# Serve two customers
queue.popleft()  # removes C1
queue.popleft()  # removes C2

# Who is now in front?
print("Person now in front of the queue:", queue[0])
```

Output (Screenshot Style):

```
Person now in front of the queue: C3
```

✅ **Answer: C3** is now at the front.

---

3. Challenge Question
Problem:
Compare **stack vs queue** for restaurant orders. Which is correct?

Algorithm Steps:

1. **Stack** works as **LIFO**: last order added would be served first.
2. **Queue** works as **FIFO**: first order added would be served first.
3. In restaurants, fairness requires **FIFO** — the first customer to order should be the first to be served.

Code Explanation (Short Demo)

```
# Challenge: Compare Stack vs Queue
stack = []                # LIFO
queue = deque()           # FIFO

# Add orders in order received
orders = ["Order1", "Order2", "Order3"]

# Add to stack & queue
for order in orders:
    stack.append(order)
    queue.append(order)

# Stack serves (pop) - LIFO
print("Stack serves:", stack.pop())  # serves Order3 first

# Queue serves (popleft) - FIFO
print("Queue serves:", queue.popleft())  # serves Order1 first
```

Output (Screenshot Style):

```
Stack serves: Order3
Queue serves: Order1
```

✅ **Answer: For restaurant orders, queue (FIFO) is correct**

---

4. Reflection Question

Question: Why is FIFO essential in health services like CHUK?

Answer (Theory Only):

FIFO (First In, First Out) is essential in health services because it ensures fairness and efficiency:

- Patients are treated in the order they arrive, preventing line-jumping.
- It reduces waiting-time complaints and maintains order.
- It helps in prioritizing resources properly and avoiding confusion.
- It reflects ethical practice, especially in public hospitals like CHUK, where fairness and transparency are crucial.