BIT - Data Structure Exercise - Number 2

Part I – STACK

A. Basics

Q1: How does this show the LIFO nature of stacks?
In the MTN MoMo app, when you fill payment details step-by-step, pressing back removes the last step you entered. This shows the Last-In-First-Out (LIFO) nature because the most recent action is undone first, just like in a stack where the last pushed element is popped out first.
Example in Python:

```
stack = []
stack.append('Step1')
stack.append('Step2')
stack.pop()   # Removes 'Step2'
```

Q2: Why is this action similar to popping from a stack?
In UR Canvas, when navigating through modules, pressing back undoes the last action. This is similar to popping in a stack because the most recent operation is removed first, just like removing the top element of a stack.
Example in Python:

```
modules = ['Intro', 'Lesson1', 'Lesson2']
modules.pop()  # Removes 'Lesson2'
```

B. Application

Q3: How could a stack enable the undo function when correcting mistakes?
A stack stores actions step by step. If a mistake occurs, the undo function simply pops the latest action from the stack, restoring the system to its previous state. This makes stacks ideal for implementing undo operations in applications.
Example in Python:

```
actions = []
actions.append('Typed A')
actions.append('Typed B')
actions.pop()   # Undo last action
```

Q4: How can stacks ensure forms are correctly balanced?
In Irembo registration forms, stacks can be used to check for balanced entries like brackets or matched fields. When an opening field is entered, it is pushed to the stack. When its corresponding closing field is detected, it is popped. If the stack is empty at the end, it means the form fields are properly balanced.

Example in Python:

```
def check_balance(expr):
  stack = []
  for ch in expr:
    if ch == '(': stack.append(ch)
    elif ch == ')':
      if not stack: return False
      stack.pop()
  return not stack
print(check_balance('(a+b)'))  # True
```

C. Logical

Q5: Which task is next (top of stack)?
The sequence is: Push("CBE notes"), Push("Math revision"), Push("Debate"), Pop(), Push("Group assignment"). After these operations, the stack contains [CBE notes, Math revision, Group assignment]. The top of the stack is "Group assignment".

Q6: Which answers remain in the stack after undoing?
If a student undoes 3 recent actions, it means 3 pops are performed on the stack. Only the earlier answers remain in the stack, while the latest three are removed. This demonstrates how multiple pops can reverse several actions.

D. Advanced Thinking

Q7: How does a stack enable this retracing process?
In RwandAir booking, every step is pushed to a stack. When a passenger goes back, the system pops the last step. This allows retracing in reverse order, step-by-step, which is exactly how stacks function.

Q8: Show how a stack algorithm reverses the proverb.
To reverse 'Umwana ni umutware':
- Push 'Umwana', Push 'ni', Push 'umutware'.
- Then Pop each word: 'umutware', 'ni', 'Umwana'.
Example in Python:

```
words = 'Umwana ni umutware'.split()
stack = []
for w in words:
  stack.append(w)
while stack:
  print(stack.pop(), end=' ')
```

Q9: Why does a stack suit this case better than a queue?
In Depth-First Search (DFS) at the Kigali Public Library, the search goes deep into one branch before backtracking. A stack is suitable because it allows returning to the most recent node explored, unlike a queue which follows a breadth-first approach.
Example in Python DFS:

```
graph = { 'A': ['B','C'], 'B':['D'], 'C':[], 'D':[] }
stack = ['A']
visited = []
while stack:
   node = stack.pop()
   if node not in visited:
      visited.append(node)
      stack.extend(graph[node])
print(visited)
```

Q10: Suggest a feature using stacks for transaction navigation.
In the BK Mobile app, a feature could allow users to move forward and backward through their transaction history. Each new transaction page is pushed onto a stack, and pressing back pops the most recent one, making navigation simple and efficient.

Part II – QUEUE

A. Basics

Q1: How does this show FIFO behavior?
In a Kigali restaurant, customers are served in the same order they arrived. This is First-In-First-Out (FIFO), just like in a queue where the first enqueued item is the first to be dequeued.
Example in Python:

```
from collections import deque
queue = deque()
queue.append('Customer1')
queue.append('Customer2')
print(queue.popleft())  # Customer1
```

Q2: Why is this like a dequeue operation?
In a YouTube playlist, the next video plays automatically. This is like dequeueing the first video in the queue, while the others wait for their turn.

B. Application

Q3: How is this a real-life queue?
At RRA offices, people line up to pay taxes. The first person in the line is served first, which

is a direct example of FIFO queue behavior.

Q4: How do queues improve customer service?
In MTN or Airtel service centers, SIM replacement requests are processed in order of arrival. This ensures fairness and prevents skipping, which improves customer satisfaction.

C. Logical

Q5: Who is at the front now?
Sequence: Enqueue(Alice), Enqueue(Eric), Enqueue(Chantal), Dequeue(), Enqueue(Jean). After these operations, the queue is [Eric, Chantal, Jean]. The front is 'Eric'.

Q6: Explain how a queue ensures fairness.
In RSSB pension applications, requests are handled in arrival order. This ensures that no one jumps the line, maintaining fairness for all applicants.

D. Advanced Thinking

Q7: Explain how each maps to real Rwandan life.
- Linear Queue: People waiting at a wedding buffet, where the first person is served first.
- Circular Queue: Buses at Nyabugogo loop through a route and return to the starting point, rejoining the queue.
- Deque: Boarding a bus where passengers can enter from both the front and rear doors.

Q8: How can queues model this process?
In a Kigali restaurant, customers place food orders which are enqueued. When the food is ready, the order is dequeued and served. This models how queues manage order and service.

Q9: Why is this a priority queue, not a normal queue?
At CHUK hospital, emergencies are treated first regardless of arrival time. This is a priority queue because patients are served based on urgency, not on FIFO order.
Example in Python:

```
import heapq
pq = []
heapq.heappush(pq, (1, 'Emergency'))
heapq.heappush(pq, (3, 'Normal Case'))
heapq.heappush(pq, (2, 'Serious Case'))
print(heapq.heappop(pq))  # (1, 'Emergency')
```

Q10: How would queues fairly match drivers and students?
In a moto/e-bike taxi app, drivers are enqueued as available, and students are enqueued as riders. When a match occurs, the driver at the front is paired with the student at the front, ensuring fairness and order in service.