PROJECT 85

- ✓ ***QUEUE ANSWERS BOTH FOR CHALLENGES AND REFLECTIONS***

1. CHALLENGE ANSWER

Queue vs stack for graduation ceremony entry. Which is proper?

For a graduation ceremony, attendees should enter in the order they arrive (first to arrive enters first) to ensure fairness. A queue (FIFO: First-In-First-Out) preserves this order, while a stack (LIFO) allows the last to arrive to enter first, which is unfair. Without loops, functions, or libraries, I'll use a Python list to simulate both a queue and a stack for three attendees (A1, A2, A3), manually performing enqueue/dequeue (for queue) and push/pop (for stack) operations. For the queue, I'll use append to add elements and pop(0) to remove the first element (FIFO). For the stack, I'll use append and pop() (LIFO).

Algorithmic Sequence:

Model Attendees: Represent three attendees (A1, A2, A3) arriving in that order.

Simulate Queue:

1. *Initialize an empty list as the queue.*
2. *Manually append A1, A2, A3 to the queue (enqueue).*
3. *Manually remove the first element (using pop(0)) three times to simulate entry order.*

Simulate Stack:

1. *Initialize an empty list as the stack.*
2. *Manually append A1, A2, A3 to the stack (push).*
3. *Manually remove the last element (using pop()) three times to simulate entry order.*

Compare and Conclude: Store entry orders in lists. The queue yields [A1, A2, A3] (fair), while the stack yields [A3, A2, A1] (unfair). Conclude that the queue is proper.

1: Simulate queue for entry

Step1: Initialize empty queue (list)

queue = []

Step 2: Manually enqueue attendees

queue.append("A1") # Queue: ['A1']

queue.append("A2") # Queue: ['A1', 'A2']

queue.append("A3") # Queue: ['A1', 'A2', 'A3']

step 3: Manually dequeue to get entry order

queue_entry1 = queue.pop(0) # Remove A1

queue_entry2 = queue.pop(0) # Remove A2

queue_entry3 = queue.pop(0) # Remove A3

step 4:Store queue entry order

queue_entry_order = [queue_entry1, queue_entry2, queue_entry3]

2: Simulate stack for entry

Step 1: Initialize empty stack (list)

stack = []

step 2: Manually push attendees

stack.append("A1") # Stack: ['A1']

stack.append("A2") # Stack: ['A1', 'A2']

stack.append("A3") # Stack: ['A1', 'A2', 'A3']

step 3: Manually pop to get entry order

stack_entry1 = stack.pop() # Remove A3

stack_entry2 = stack.pop() # Remove A2

stack_entry3 = stack.pop() # Remove A1

step 4: Store stack entry order

stack_entry_order = [stack_entry1, stack_entry2, stack_entry3]

3: Print results

print("Entry order (Queue):", queue_entry_order)

print ("Entry order (Stack):", stack_entry_order)

2. Reflection answer

Queue Reflection: Why FIFO matches fairness in ceremonies?

The First-In-First-Out (FIFO) principle, which defines a queue, ensures that items or people are processed in the exact order they arrive. This aligns perfectly with fairness in ceremonies, such as a graduation event, where attendees expect to be admitted or seated based on their arrival time. Fairness in such contexts is tied to the principle that those who arrive early, investing more time and effort, should be rewarded with earlier entry or service. For instance, if students or guests line up for a graduation ceremony, the first to arrive should enter first, as this respects their punctuality and maintains an orderly process.

FIFO's fairness is evident in its predictability and transparency. In a ceremony, if later arrivals were allowed to enter before earlier ones, it would create resentment and disrupt the event's structure, potentially causing chaos in large gatherings. A queue ensures a smooth, equitable flow, as seen in scenarios like hospital patient lines or telecom service counters, where the first person in line is served first. This orderly progression prevents disputes and aligns with societal expectations of justice in organized events. By contrast, a LIFO approach, like a stack, would allow latecomers to take precedence, violating fairness and undermining the ceremonial experience. FIFO's alignment with arrival order makes it the ideal choice for ensuring equity and maintaining harmony in ceremonies.

END