Queue and stack operations

Stack Q3.

Algorithm: push and pop operation

1. Initialize an empty stack

Stack= [] this line initializes an empty stack that will store our elements

2. Push element onto the stack

Stack = [‘1’,’2’,’3’,’4’] this line pushes the elements [‘1’,’2’,’3’,’4’] into the stack. In a real-world scenario, this could be done using loop

3. Pop an element from the stack

Stack. Pop () this line removes the new top element (‘3’) from the stack. The stack now become [‘1’,’2’,’3’]

4. Pop another element from the stack

Stack. Pop () this line prints the remaining elements in the stack, which are [‘1’,’2’]

5. Print the remaining elements

Print (stack) this line prints the remaining elements in the stack, which are [‘1’,’2’]

Q4.**why stack are unsuitable for queue in daily life**

Stack and queue are both data structures that follow specific rules for adding and removing element. However, they differ in their ordering principles:

1. Stacks: Follow the Last-In-First-out (LIFO) principle, where the last element added is the first one to be removed.

2. Queue: follow the First-In-Last-Out (FIFO) principle, where the first element added is the first one to be removed.

In daily life queues are often used to manage tasks, request, or people in fair and orderly manner. Stack, on the other hand, are not suitable for queues in daily life because:

.unfair ordering: stack would prioritize the most recent additions, which could lead to unfair treatment of those who have been waiting longer.

.In efficient management: stacks would require constant reordering to ensure fairness, which could lead to in efficiencies and confusion.

Example: a bank teller line where customers are served based on who arrived last (LIFO), instead of who arrived first (FIFO), would be unfair and inefficient. Customers who arrived earlier would have to wait longer, leading to dissatisfaction.

Queue operations

Q3.**queue vs stack for concert entry**

Algorithm sequence

1. Arrival of people

.the people arrive at concert venue and want to enter the concert.

2. Queue approach

.people from line (queue) and enter the concert in the order they arrived (FIFO).

.queue = [] (initialize an empty queue)

.queue. Append (person) (add person to the end of queue)

.queue. Pop (0) (remove person from the front of the queue)

3. Stack approach

.people enter the concert in the reverse order they arrived (LIFO)

.stack = [] (initialize an empty stack)

.stack. Append (person) (add person to the top of the stack)

.stack. Pop () (remove person from the top of the stack)

Q4.FIFO keeps order at public event by:

.preserving temporal order: FIFO maintains the temporal order of arrival, ensuring that individuals who arrive first served first. This preserve the natural order of events and prevents chaos.

.Ensuring fairness and equity: FIFO ensure that everyone is treated equally and fairly, regardless of their social status, wealth, or influence. It eliminates favoritism and bias, promotion a sense of justice and equality.

.providing predictability and transparency: FIFO enable efficient allocation of resources, such as seats, tickets or services. By serving individuals in the order they arrive, resources are utilized optimally, minimizing waste and maximizing capacity.

.Enabling efficient resource allocation

.reducing conflict: FIFO minimizing conflict and disputes that may arise from perceived unfairness or favoritism. By following a clear and consistent order, individual are less likely to feel cheated or mistreated.

This result in a smooth, enjoyable, and fair experience for all participants.

Example:

At concert, people waiting in line to buy tickets are served in the order they arrived (FIFO). This ensure that:

.those who arrived first get tickets first

.everyone is treated fairly and equally

.the line moves in a predictable and orderly manner

This makes the experience more enjoyable and fair for all applicants.