H8_Hands_on

1. Leverage your implementation of quicksort to implement the ith order statistic. Demonstrate it's working via an example

   ⇨ Solution:

Here,

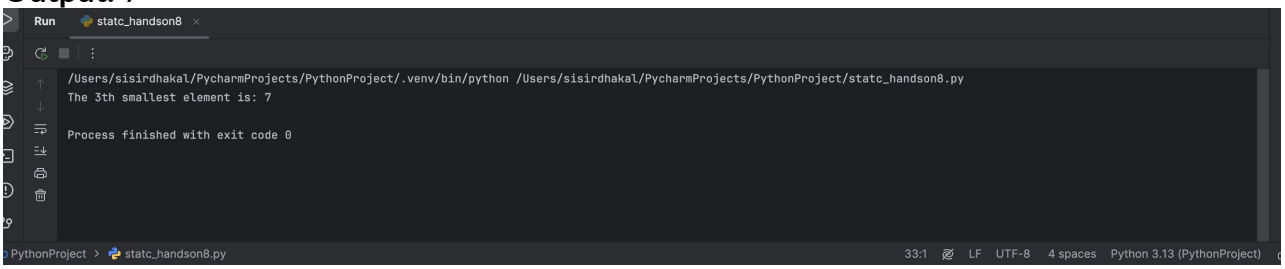
⇨ **Implementation Steps**

- Select a **pivot** and partition the array such that elements less than the pivot go to the left and those greater go to the right.
- Determine the **position of the pivot** after partitioning.
- If the pivot's position is **i**, return it.
- If **i** is smaller, recurse on the **left partition**.
- If **i** is larger, recurse on the **right partition**.

⇒ **Step-by-step Execution**

- **Initial array**: [7, 10, 4, 3, 20, 15]
- **Partition around pivot (15)**: [7, 10, 4, 3, 15, 20] → Pivot index = **4**
  - Since i=2 is smaller, recurse left.
- **Partition left [7, 10, 4, 3] around pivot (3)**: [3, 4, 7, 10] → Pivot index = **0**
  - Since i=2 is greater, recurse right.
- **Partition [4, 7, 10] around pivot (7)**: [4, 7, 10] → Pivot index = **2**
  - Found the **3rd smallest element: 7**.

⇒ **Output:** 7



⇨ This algorithm runs in **O(n) on average**, much faster than sorting.

---

2. Implement and upload your source code to github for: stack, queue, and singly linked list. Make sure to implement the same functionality (api/interface) as the ones from the book. *Restriction*: Use fixed sized arrays (C style arrays) and assume only integers (C style int) for the data to store.

⇒ Solution:
Here,
Implementation of a **Stack**, **Queue**, and **Singly Linked List** using **fixed-size arrays** (C-style) and storing only integers

⇨ Stack (Fixed-Size Array)

- **push(int x)** → Add element to the top.
- **pop()** → Remove top element.
- **top()** → Get the top element.
- **isEmpty()** → Check if stack is empty.
- **isFull()** → Check if stack is full.

⇨ Queue (Fixed-Size Array)

- **enqueue(int x)** → Add element to the rear.
- **dequeue()** → Remove front element.
- **front()** → Get the front element.
- **isEmpty()** → Check if queue is empty.
- **isFull()** → Check if queue is full.

⇨ Singly Linked List (Fixed-Size Array)

- **insert(int x)** → Insert at the end.
- **remove(int x)** → Remove first occurrence.
- **display()** → Print all elements.

Output

```
Stack top: 30
Stack top after pop: 20
Queue front: 1
Queue front after dequeue: 2
Linked List: 5 -> 10 -> 15 -> NULL
After Removing 10: 5 -> 15 -> NULL


=== Code Execution Successful ===
```