Set A:

**public static int getMaxConsecutiveOccurrence(int[] arr) {**

**if (arr == null || arr.length == 0) {**

**return 0; // empty array returns 0 (2)**

**}**

**int maxCount = 1;**

**int currentCount = 1;**

**for (int i = 1; i < arr.length; i++) {**

**if (arr[i] == arr[i - 1]) { // comparison with prev item (5)**

**currentCount++;**

**} else {**

**maxCount = Math.max(maxCount, currentCount); // update count**

**currentCount = 1; // reset count (5)**

**}**

**}**

**return Math.max(maxCount, currentCount); // Last check(2)**

**}**

| Task | Marks |
| --- | --- |
| An empty array returns 0 | 3 |
| Comparison of item with previous item | 5 |
| Resetting count to 1 when mismatch | 5 |
| Last check of currentCount with maxCount | 2 |

Set B:

If someone uses the nodeAT() function, the solution is exactly the same as the previous one.

**public static int getMaxConsecutiveOccurrence(ListNode head) {**

**if (head == null) {**

**return 0; // Return 0 for an empty list**

**}**

**int maxCount = 1;**

**int currentCount = 1;**

**ListNode current = head;**

**while (current.next != null) {**

**if (current.val == current.next.val) {**

**currentCount++;**

**} else {**

**maxCount = Math.max(maxCount, currentCount);**

**currentCount = 1;**

**}**

**current = current.next;**

**}**

**return Math.max(maxCount, currentCount); // Final check**

**}**

| Task | Marks |
| --- | --- |
| An empty linked list returns 0 | 3 |
| Comparison of current node with next node | 5 |
| Resetting count to 1 when mismatch | 5 |
| Last check of currentCount with maxCount | 2 |