Question 1: During gameplay, you will add and remove keys to the board. What kind of operations would that mean? Please elaborate.

Adding and removing keys involve the following operations:

1. **Adding Keys:**
    a. **Insert Operation:** A key is added to the board at a specific position or appended at the end. In an array, you check for the next empty slot to add the key. In a linked list, you create a new node and adjust pointers.
    b. **Sorting After Addition:** After a key is added, the keys may need to be sorted to form valid blocks or pairs.
2. **Removing Keys:**
    a. **Delete Operation:** A key is removed from the board, which may require shifting elements in an array to fill the gap. In a linked list, you adjust the pointers to skip over the removed node.
    b. **Rearranging Blocks After Deletion:** After removing a key, blocks must be re-evaluated to ensure they remain valid (e.g., size ≥ 3 for blocks or forming valid pairs).

Question 2: To determine if a user is done, what kinds of checks would you need to do? Please elaborate.

To check if a user is "done," we need to validate their 14 keys based on the two winning conditions:

1. Validation of Blocks (Condition 1):

Definition:

Keys must form groups of consecutive numbers of the same color (e.g., 2,3,4 in red) or the same number with different colors (e.g., three 11s in different colors).

Steps:

Sort Keys: Sort keys by color and number for easier grouping.

Group Consecutive Numbers: Traverse sorted keys and check for runs of consecutive numbers with the same color.

Group Same Number, Different Colors: Group keys with the same number but different colors.

Check Block Sizes: Ensure each group/block has a size of 3 or more.

Complexity:

Sorting: □(□log□)O(nlogn).

Grouping: □(□)O(n).

2. Validation of Pairs (Condition 2):

Definition:

Keys must form 7 pairs, where a pair is defined as two keys with the same number and same color.

Steps:

Count Duplicates: Iterate through keys to count duplicates of each number-color combination.

Check for Exactly 7 Pairs: Ensure there are exactly 7 pairs (each pair has a count of 2).

Complexity:

Counting: □(□)O(n).

3. Handling Errors or Invalid Cases: If keys do not satisfy either condition, the user is not "done."

Errors include: Less than 14 keys. Invalid blocks (size < 3). Invalid pair count (not exactly 7 pairs).

Question 3: Would you rather hold the 14 keys in a single fixed-size Java array? Or would you have multiple arrays or linked lists to hold the blocks? Please elaborate.

Option 1: Single Fixed-Size Array

Advantages:

Simplicity: Managing a single array is straightforward. Easy to sort and validate the keys.

Efficient Access: Random access allows efficient key retrieval (□(1)O(1)).

Low Memory Overhead: A single array has less overhead compared to linked lists.

Disadvantages:

Rigid Structure: Rearranging blocks (e.g., splitting or merging) requires significant shifting. Adding/removing keys is less flexible.

No Dynamic Expansion: Fixed-size arrays cannot accommodate more than 14 keys.

Option 2: Multiple Arrays

Advantages:

Logical Grouping: Each block can have its own array, making grouping and validation easier.

For example:

Array 1: [2,3,4] (same color, consecutive numbers)

Array 2: [11,11,11] (same number, different colors)

Modularity: Each block can be processed independently.

Flexibility: Blocks can grow or shrink independently of other blocks.

Disadvantages:

Overhead: Managing multiple arrays requires more memory and logic.

Access Complexity: Keys are scattered across arrays, making traversal more complex.

Option 3: Linked Lists

Advantages:

Dynamic Structure: Keys can be easily added or removed without shifting elements. Blocks can be represented as linked lists of keys.

Efficient Merging: Blocks can be combined by adjusting pointers.

Disadvantages:

Sequential Access: Accessing specific keys is slower (□(□)O(n)).

Sorting: Sorting linked lists is more complex compared to arrays.

Recommendation: The choice depends on the specific requirements:

If simplicity and performance are the main goals: Use a single fixed-size array for the 14 keys.

If flexibility in managing blocks is important: Use multiple arrays or linked lists to represent blocks dynamically.

Final Summary:

Operations for Adding/Removing Keys: Insert and delete operations, along with sorting and revalidation of blocks.

Checks to Determine if Done: Validate blocks of size 3+ and 7 pairs.

Preferred Data Structure: A single array for simplicity or multiple arrays/linked lists for flexibility, depending on the use case.