More 1D-Array Practice Problems

Complete each of the following practice problems. You should consider edge cases and check for invalid input in each case. Write your answers on separate paper.

Even Index Elements

```java
/**
 * Returns a new array containing only the elements at even indices from the input array.
 * @param arr the input array
 * @return array containing elements at indices 0, 2, 4, etc. from input
 */
public static int[] getEvenIndexElements(int[] arr)
```

- Example: `[5,2,8,1,9,4]` → `[5,8,9]`

Adjacent Sum Pair

```java
/**
 * Finds the first pair of adjacent elements that sum to the target value.
 * @param arr the input array
 * @param target the target sum to find
 * @return array containing indices of the pair, or [-1,-1] if no such pair exists
 */
public static int[] findAdjacentSum(int[] arr, int target)
```

- Example: `([1,3,5,2,7,5], target=7)` → `[2,3]` (since `arr[2]+arr[3]=7`)

Longest Streak

```java
/**
 * Returns the length of the longest streak of consecutive equal values.
 * @param arr the input array
 * @return length of longest streak of equal values, including ties
 */
public static int longestStreak(int[] arr)
```

- Example: `[1,1,1,2,2,6,6,6,6]` → `4`
- Example: `[3,7,7,7,2,2]` → `3`

Balance Point

```java
/**
 * Finds index where sum of numbers to left (including index)
 * equals sum of numbers to right of the index
 * @param arr the input array
 * @return index of balance point, or -1 if none exists
 */
public static int findBalancePoint(int[] arr)
```

- Example: `[1,2,3,6]` → `2` (since $1+2+3 = 6$)
- Example: `[5,2,6]` → `-1`

Alternating Signs

```java
/**
 * Checks if array alternates between positive and negative numbers, starting with positive
 * @param arr the input array
 * @return true if signs alternate and no zeros present, false otherwise
 */
public static boolean hasAlternatingSigns(int[] arr)
```

- Example: `[1,-3,2,-4,5] → true`
- Example: `[1,-3,2,-4,0] → false`

Maximum Sliding Window

```java
/**
 * Finds largest sum of k consecutive elements in the array.
 * @param arr the input array
 * @param k the window size
 * @return largest sum of k consecutive elements, or -1 if k > array length
 */
public static int maxSlidingWindow(int[] arr, int k)
```

- Example: `([1,4,2,7,3,1], k=3) → 13 (4+2+7)`
- Example: `([1,2,3], k=4) → -1`

Harder Problems

Mountain Sequence

```java
/**
 * Determines if array contains a mountain sequence (increases then decreases).
 * Sequence must be at least 3 numbers long.
 * @param arr the input array
 * @return true if array contains a mountain sequence, false otherwise
 */
public static boolean isMountain(int[] arr)
```

- Example: `[1,4,6,4,2] → true`
- Example: `[1,2,3] → false`
- Example: `[5,2,1] → false`

Most Frequent Element

```java
/**
 * Returns most frequently occurring element. If tie, returns first occurrence.
 * @param arr the input array
 * @return most frequent element in the array
 */
public static int findMode(int[] arr)
```

- Example: `[1,2,2,3,3,3,4] → 3`
- Example: `[1,2,2,1] → 1`

Range Compression

```java
/**
 * Converts sorted array into string showing ranges of consecutive numbers.
 * @param arr the input sorted array
 * @return string representation of ranges
 */
public static String compressRanges(int[] arr)
```

- Example: `[1,2,3,5,6,7,9] → "1-3,5-7,9"`
- Example: `[1,2,4,7,8] → "1-2,4,7-8"`