DATE: 13-11-2024

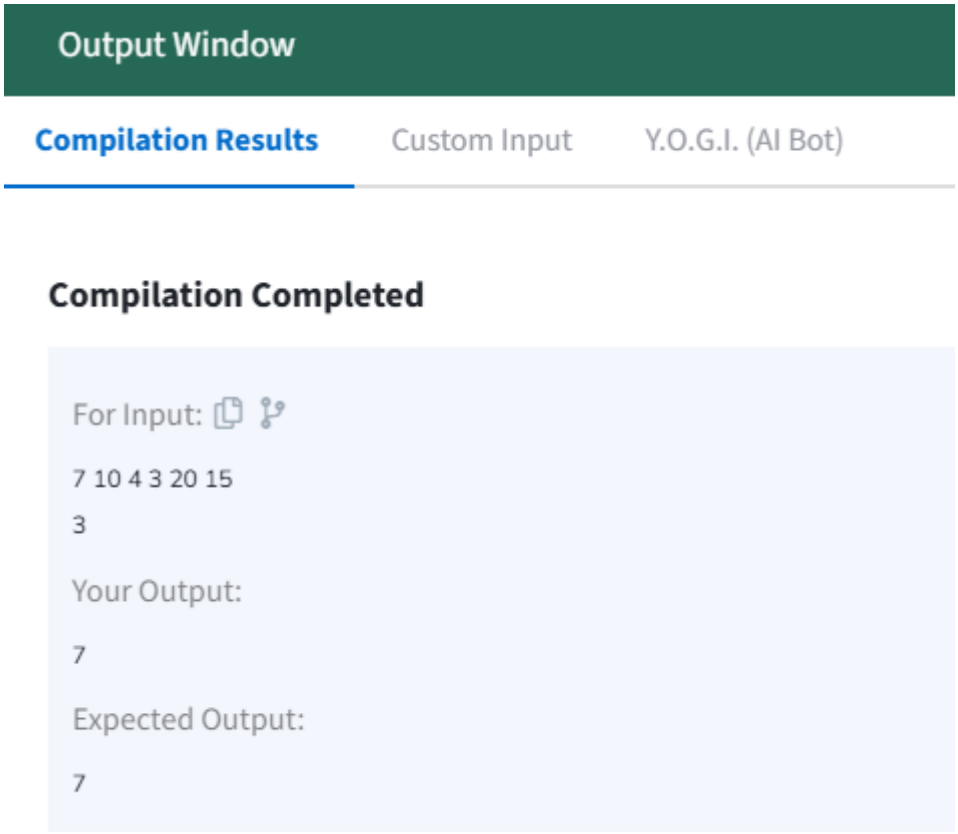
CODING PRACTICE PROBLEMS – DAY 4

1. Kth Smallest

```java
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int num : arr) {
            minHeap.add(num);
        }
        for (int i = 1; i < k; i++) {
            minHeap.poll();
        }
        return minHeap.poll();
    }
}
```

Time complexity: O(n + k log n)

OUTPUT:

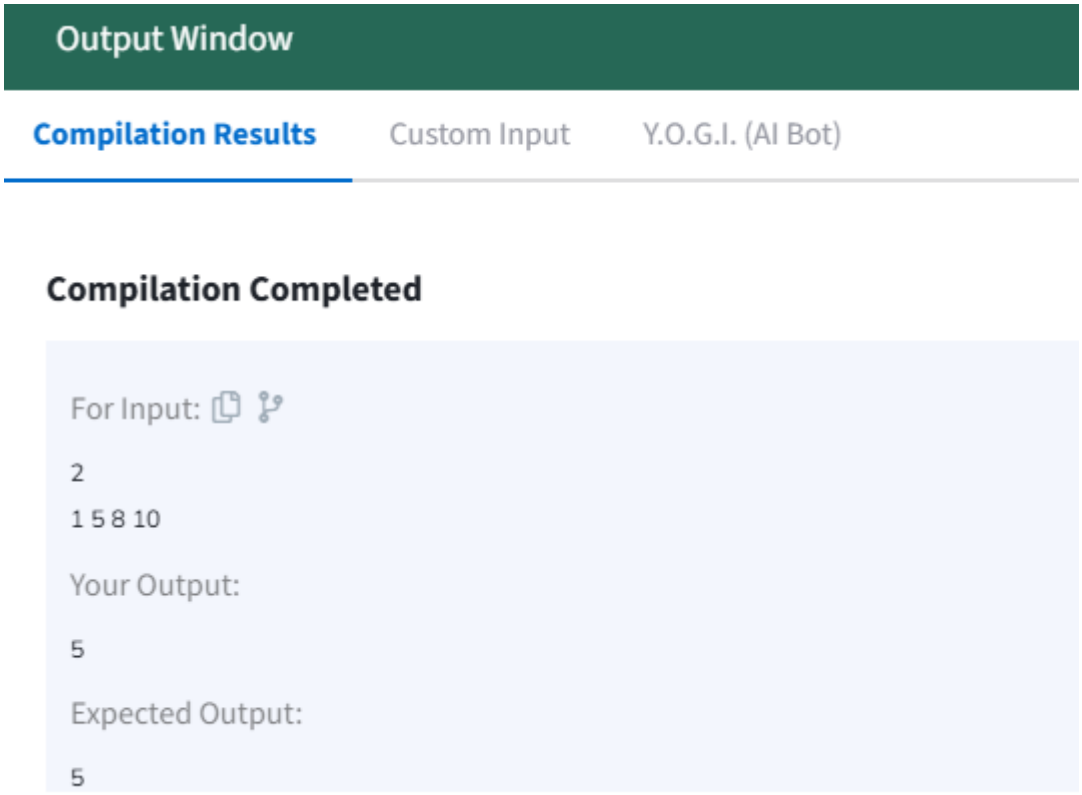


2. Minimize the Heights II

```java
class Solution {
    int getMinDiff(int[] arr, int k) {
        int n = arr.length;
        Arrays.sort(arr);
        int ans = arr[n-1] - arr[0];
        for (int i = 0; i < n - 1; i++) {
            int minHeight = Math.min(arr[0] + k, arr[i+1] - k);
            int maxHeight = Math.max(arr[n-1] - k, arr[i] + k);

            if (minHeight < 0) continue;

            ans = Math.min(ans, maxHeight - minHeight);
        }

        return ans;
    }
}
```

Time complexity: O(n log n)

OUTPUT:

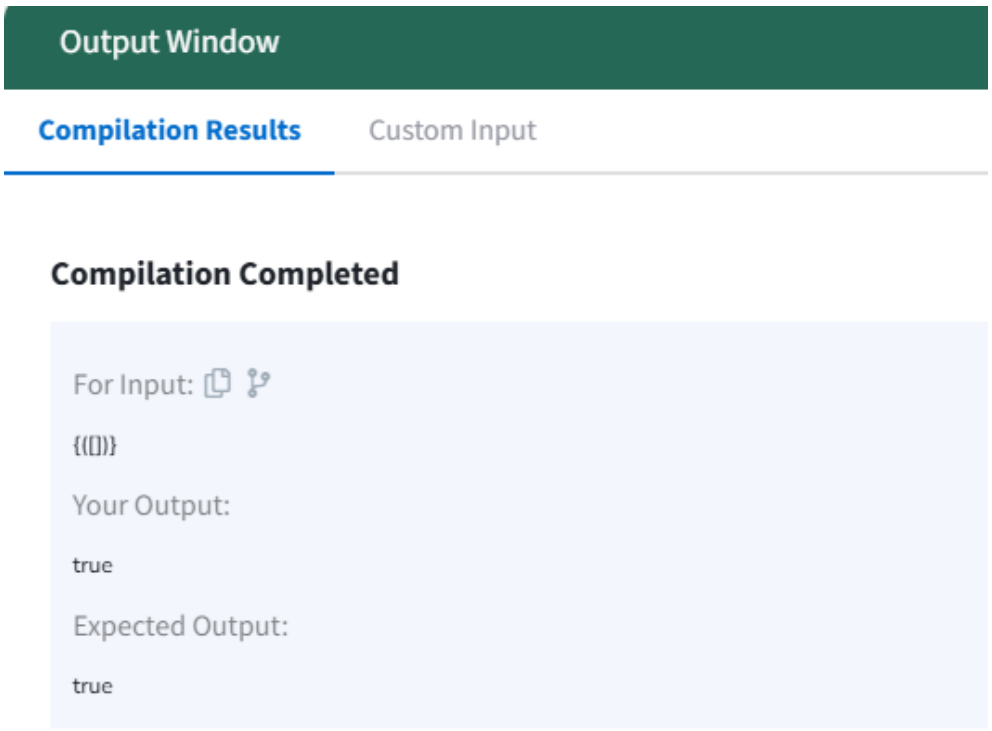


3. Parenthesis Checker

```
class Solution {
    static boolean isParenthesisBalanced(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '{' || c == '(' || c == '[') {
                stack.push(c);
            }
            else if (c == '}' || c == ')' || c == ']') {
                if (stack.isEmpty()) {
                    return false;
                }
                char top = stack.pop();
                if ((c == '}' && top != '{') ||
                    (c == ')' && top != '(') ||
                    (c == ']' && top != '[')) {
                    return false;
                }
            }
        }
        return stack.isEmpty();
    }
}
```

Time complexity: O(n)

OUTPUT:

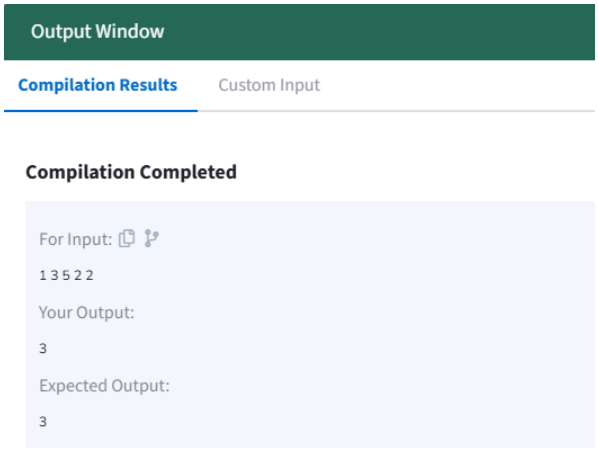


4. Equilibrium Point

```java
class Solution {
    public static int equilibriumPoint(int[] arr) {
        int n = arr.length;
        if (n == 1) return 1;
        int totalSum = 0;
        for (int num : arr) {
            totalSum += num;
        }
        int leftSum = 0;
        for (int i = 0; i < n; i++) {
            totalSum -= arr[i];
            if (leftSum == totalSum) {
                return i + 1;
            }
            leftSum += arr[i];
        }
        return -1;
    }
}
```

Time complexity: O(n)

OUTPUT:

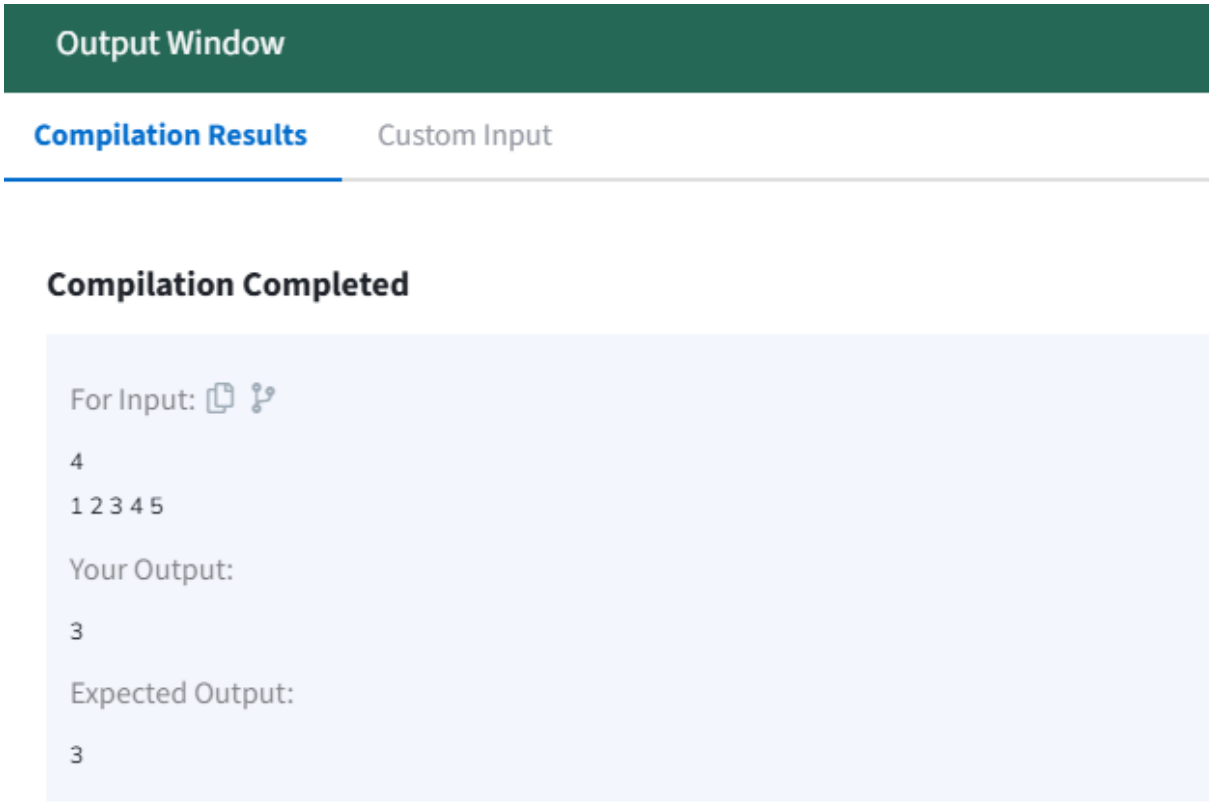


5. Binary Search

```
class Solution {
    public int binarysearch(int[] a, int k) {
        int start=0;
        int end=a.length-1;

        while(start<=end){
            int mid=start+(end-start)/2;

            if(a[mid]==k){
                return mid;
            }else if(a[mid]<k){
                start=mid+1;
            }else{
                end=mid-1;
            }
        }
        return -1;
    }
}
```

Time complexity: O(log n)

OUTPUT:

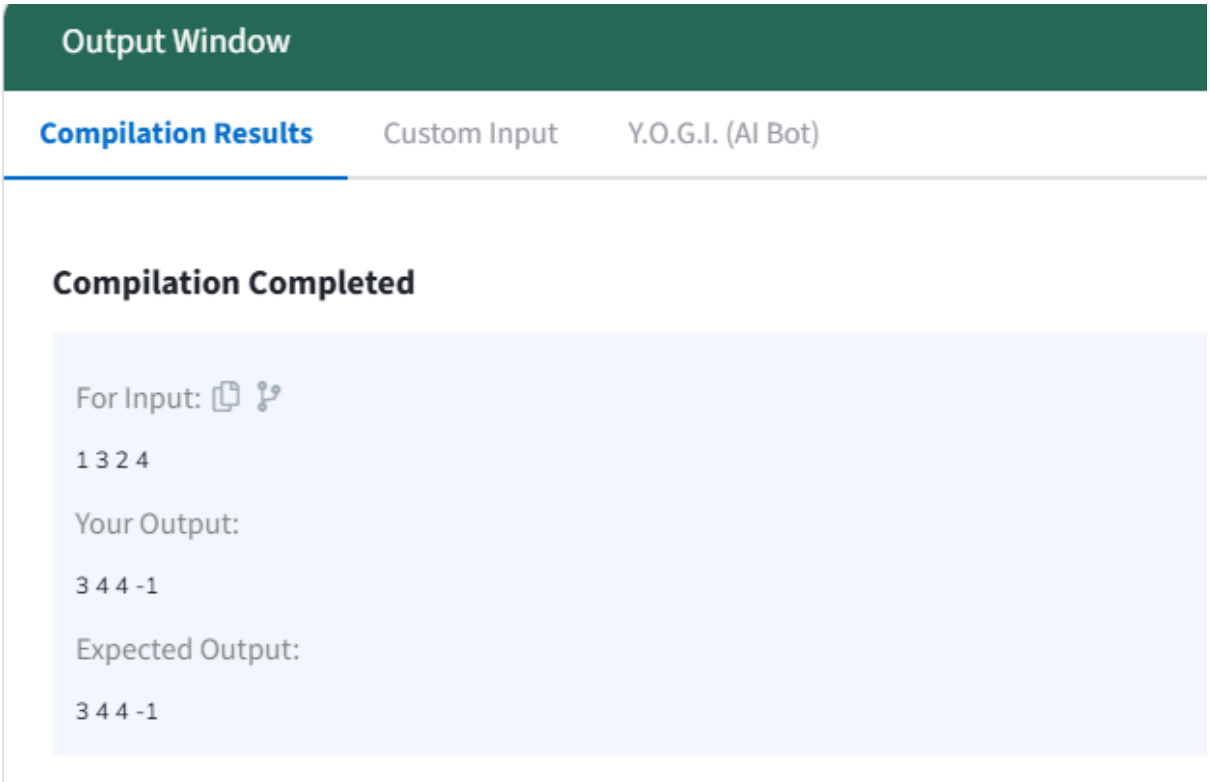


6. Next Greater Element

```java
class Solution {
    public ArrayList<Integer> nextLargerElement(int[] arr) {
        ArrayList<Integer> result = new ArrayList<>(arr.length);
        Stack<Integer> stack = new Stack<>();
        for (int i = 0; i < arr.length; i++) {
            result.add(-1);
        }
        for (int i = arr.length - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }
            if (!stack.isEmpty()) {
                result.set(i, stack.peek());
            }
            stack.push(arr[i]);
        }
        return result;
    }
}
```

Time complexity: O(n)

OUTPUT:



7. Union of Two Arrays with Duplicate Elements

```java
class Solution {
    public static int findUnion(int[] a, int[] b) {
        Set<Integer> unionSet = new HashSet<>();
        for (int num : a) {
            unionSet.add(num);
        }
        for (int num : b) {
            unionSet.add(num);
        }
        return unionSet.size();
    }
}
```

Time complexity: O(n+m)

OUTPUT: