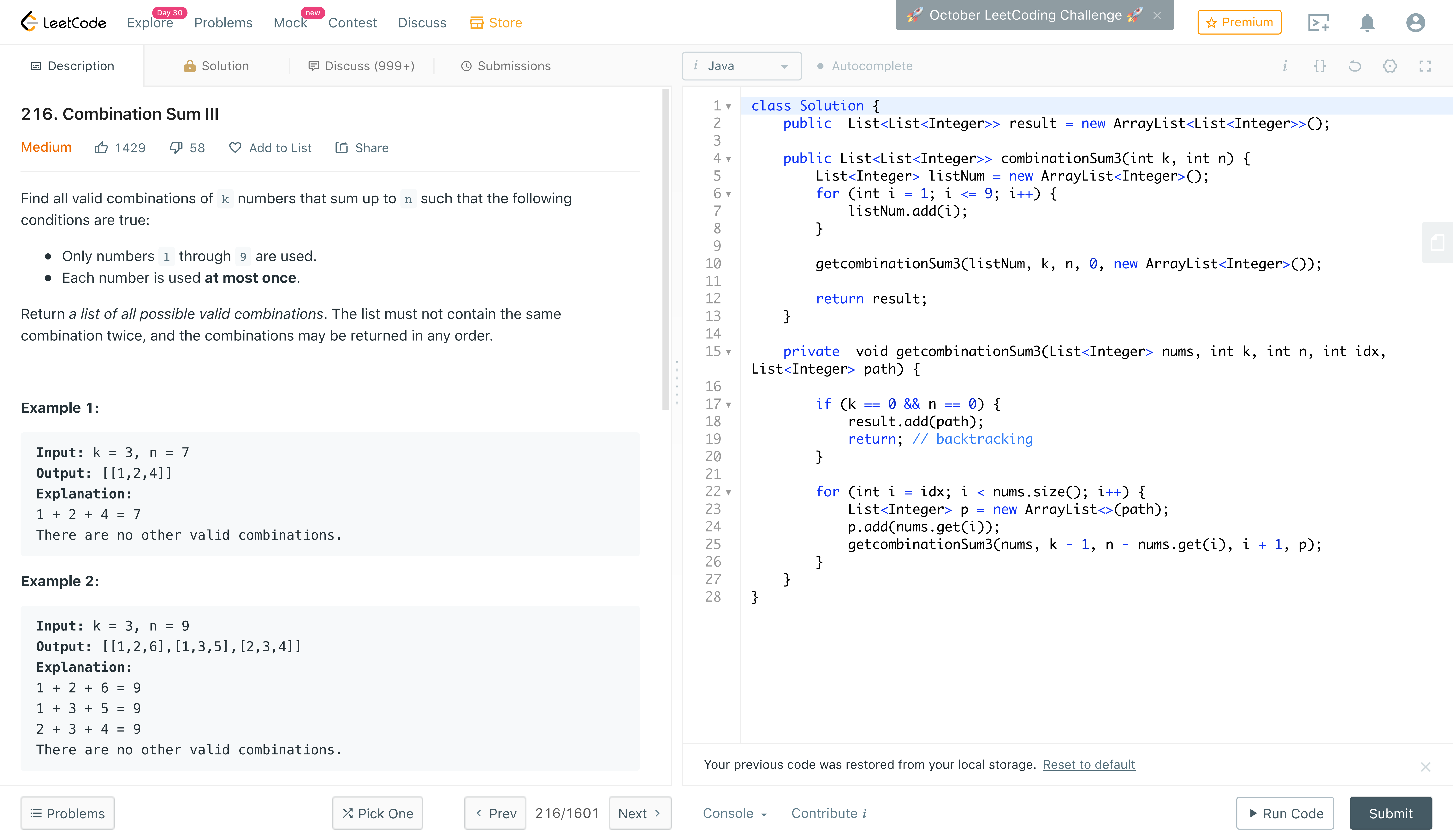

216. Combination Sum III

Medium 1429 58 Add to List Share

Find all valid combinations of `k` numbers that sum up to `n` such that the following conditions are true:

- Only numbers `1` through `9` are used.
- Each number is used **at most once**.

Return *a list of all possible valid combinations*. The list must not contain the same combination twice, and the combinations may be returned in any order.

Example 1:

```
Input: k = 3, n = 7
Output: [[1,2,4]]
Explanation:
1 + 2 + 4 = 7
There are no other valid combinations.
```

Example 2:

```
Input: k = 3, n = 9
Output: [[1,2,6],[1,3,5],[2,3,4]]
Explanation:
1 + 2 + 6 = 9
1 + 3 + 5 = 9
2 + 3 + 4 = 9
There are no other valid combinations.
```

```java
class Solution {
    public  List<List<Integer>> result = new ArrayList<List<Integer>>();

    public List<List<Integer>> combinationSum3(int k, int n) {
        List<Integer> listNum = new ArrayList<Integer>();
        for (int i = 1; i <= 9; i++) {
            listNum.add(i);
        }

        getcombinationSum3(listNum, k, n, 0, new ArrayList<Integer>());

        return result;
    }

    private  void getcombinationSum3(List<Integer> nums, int k, int n, int idx,
List<Integer> path) {

        if (k == 0 && n == 0) {
            result.add(path);
            return; // backtracking
        }

        for (int i = idx; i < nums.size(); i++) {
            List<Integer> p = new ArrayList<>(path);
            p.add(nums.get(i));
            getcombinationSum3(nums, k - 1, n - nums.get(i), i + 1, p);
        }
    }
}
```

Your previous code was restored from your local storage. Reset to default