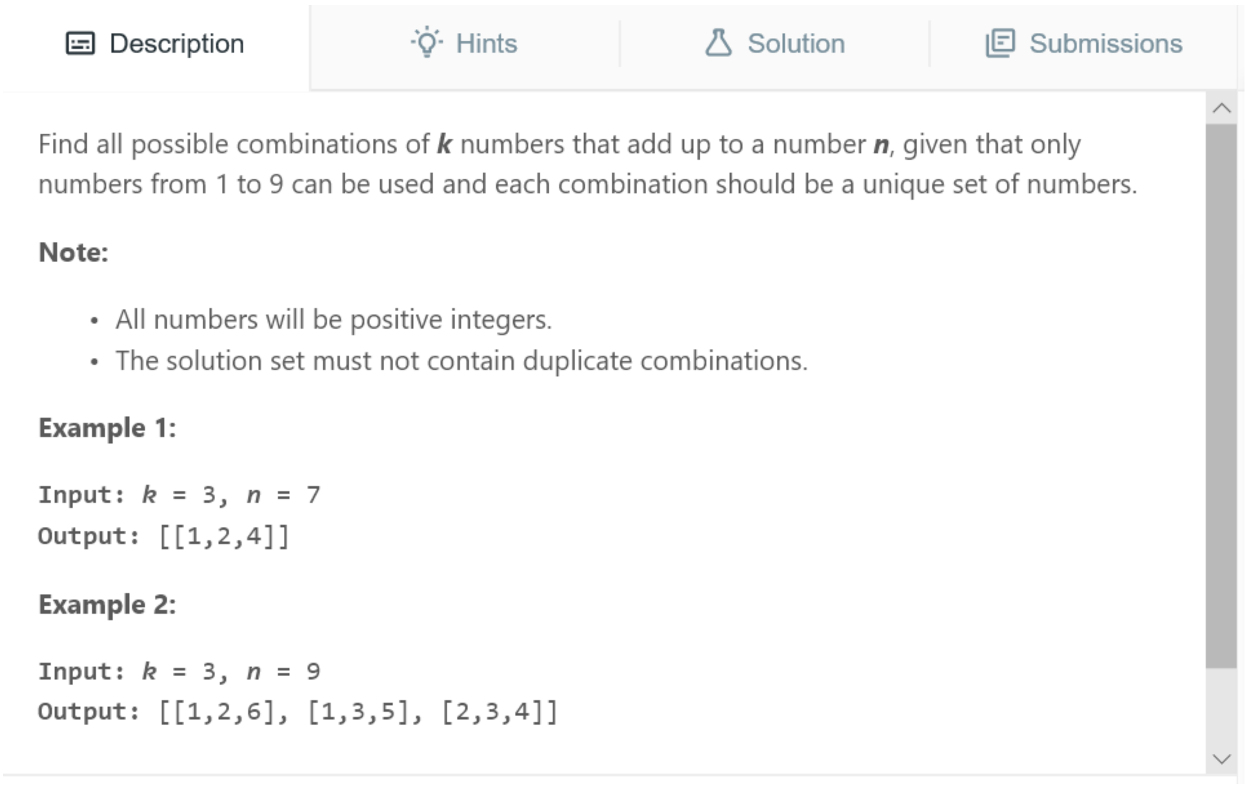


Find all possible combinations of ***k*** numbers that add up to a number ***n***, given that only numbers from 1 to 9 can be used and each combination should be a unique set of numbers.

Note:

- All numbers will be positive integers.
- The solution set must not contain duplicate combinations.

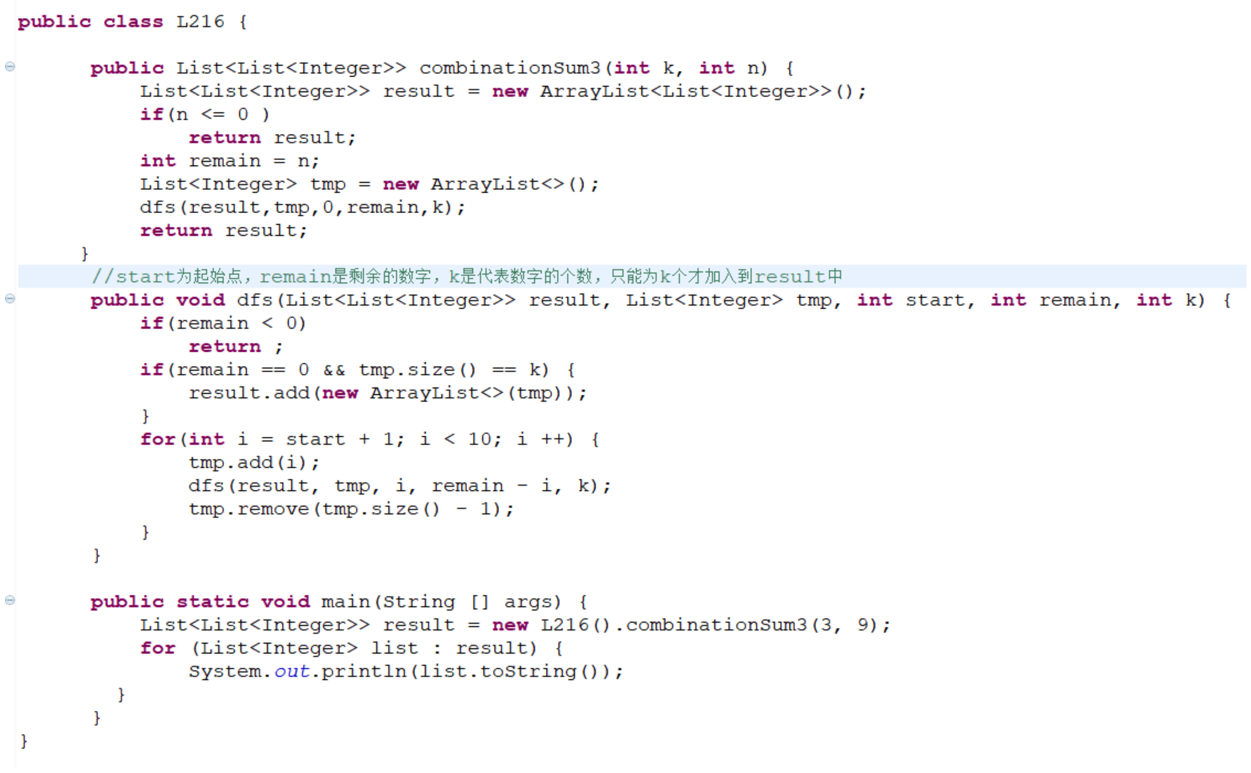
Example 1:

```
Input: k = 3, n = 7
Output: [[1,2,4]]
```

Example 2:

```
Input: k = 3, n = 9
Output: [[1,2,6], [1,3,5], [2,3,4]]
```

利用深度遍历来进行处理



```
public class L216 {

    public List<List<Integer>> combinationSum3(int k, int n) {
        List<List<Integer>> result = new ArrayList<List<Integer>>();
        if(n <= 0 )
            return result;
        int remain = n;
        List<Integer> tmp = new ArrayList<>();
        dfs(result,tmp,0,remain,k);
        return result;
    }
    //start为起始点，remain是剩余的数字，k是代表数字的个数，只能为k个才加入到result中
    public void dfs(List<List<Integer>> result, List<Integer> tmp, int start, int remain, int k) {
        if(remain < 0)
            return ;
        if(remain == 0 && tmp.size() == k) {
            result.add(new ArrayList<>(tmp));
        }
        for(int i = start + 1; i < 10; i ++) {
            tmp.add(i);
            dfs(result, tmp, i, remain - i, k);
            tmp.remove(tmp.size() - 1);
        }
    }

    public static void main(String [] args) {
        List<List<Integer>> result = new L216().combinationSum3(3, 9);
        for (List<Integer> list : result) {
            System.out.println(list.toString());
        }
    }
}
```