Experiment – 2.3

Student Name: Avinash Jena **UID: 20BCS2690**

Branch: CSE **Section/Group: 707/B**

Subject Name: Competitive Coding II **Subject Code: 20CSP-351**

Aim: Count and Say

Objective:

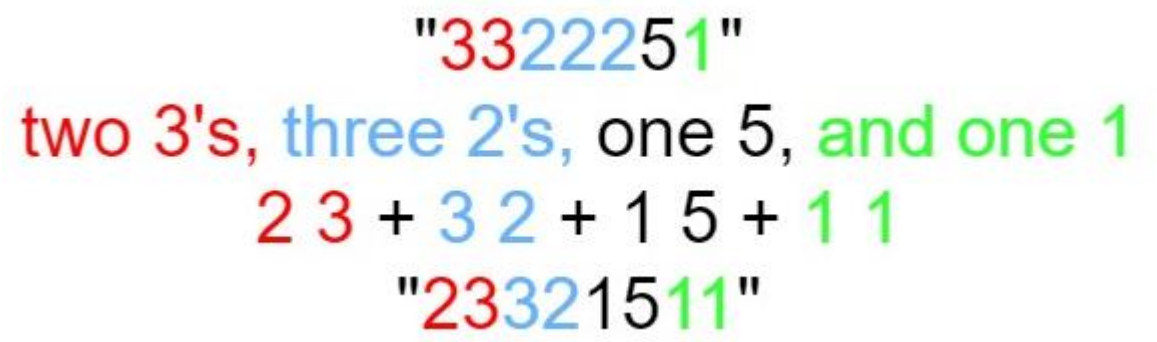
The count-and-say sequence is a sequence of digit strings defined by the recursive formula:

countAndSay(1) = "1"

countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string "3322251":



Given a positive integer n, return the nth term of the count-and-say sequence.

Example 1:

Input: n = 1

Output: "1"

Explanation: This is the base case.

Example 2:

Input: n = 4

Output: "1211"

Explanation:

countAndSay(1) = "1"

countAndSay(2) = say "1" = one 1 = "11"

countAndSay(3) = say "11" = two 1's = "21"

countAndSay(4) = say "21" = one 2 + one 1 = "12" + "11" = "1211"

Constraints:

1 <= n <= 30

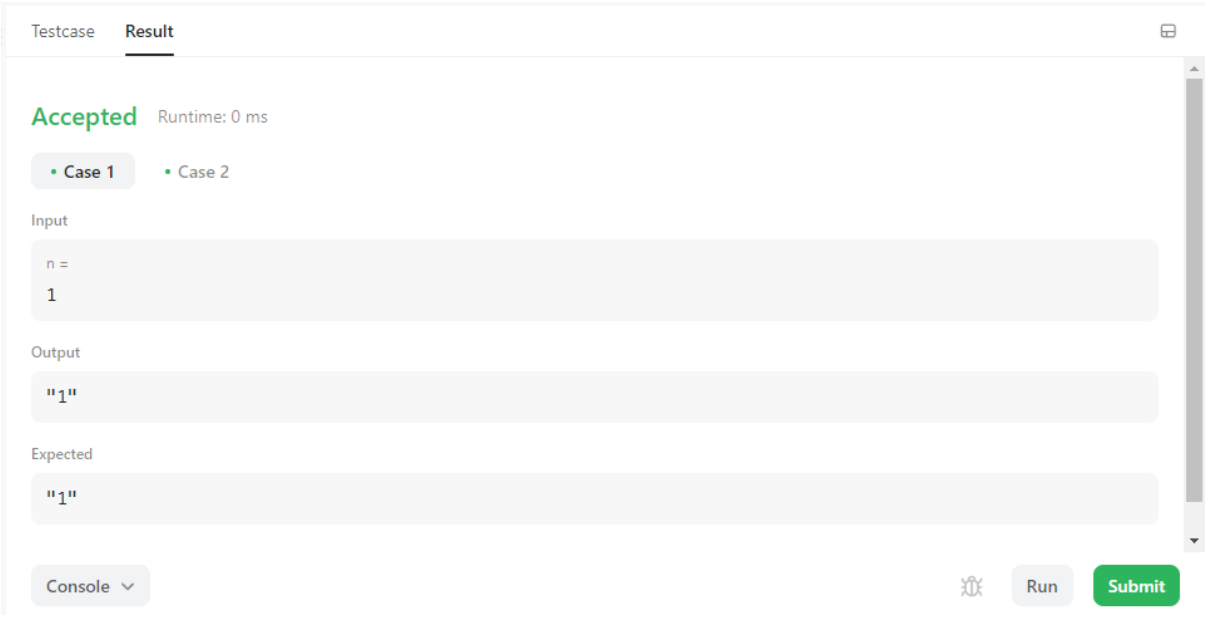
Code:

```
class Solution {
    public String countAndSay(int n) {
        if(n==1) return "1";
        String prev = countAndSay(n-1);
        StringBuilder sb = new StringBuilder();
            int numDigit = 1;
        char currentNum = prev.charAt(0);
        for(int i=1;i<prev.length();i++){
            if(prev.charAt(i) == currentNum){
                numDigit++;
            }else{
                sb.append(numDigit).append(currentNum);
                numDigit = 1;
                currentNum = prev.charAt(i);
            }
        }

        if(numDigit>0){
            sb.append(numDigit).append(currentNum);
        }
        return sb.toString();
    }
}
```

Output:

Java Auto

```java
class Solution {
    public String countAndSay(int n) {
        if(n==1) return "1";
        String prev = countAndSay(n-1);
        StringBuilder sb = new StringBuilder();
        int numDigit = 1;
        char currentNum = prev.charAt(0);
        for(int i=1;i<prev.length();i++){
            if(prev.charAt(i) == currentNum){
                numDigit++;
            }else{
                sb.append(numDigit).append(currentNum);
                numDigit = 1;
                currentNum = prev.charAt(i);
            }
        }

        if(numDigit>0){
            sb.append(numDigit).append(currentNum);
        }
        return sb.toString();
    }
}
```



Submitted By – Avinash Jena

Aim: Water and Jug Problem

Objective:

You are given two jugs with capacities jug1Capacity and jug2Capacity liters. There is an infinite amount of water supply available. Determine whether it is possible to measure exactly targetCapacity liters using these two jugs.
If targetCapacity liters of water are measurable, you must have targetCapacity liters of water contained within one or both buckets by the end.
Operations allowed:
Fill any of the jugs with water.
Empty any of the jugs.
Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty.

Example 1:
Input: jug1Capacity = 3, jug2Capacity = 5, targetCapacity = 4
Output: true
Explanation: The famous Die Hard example

Example 2:
Input: jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5
Output: false

Example 3:
Input: jug1Capacity = 1, jug2Capacity = 2, targetCapacity = 3
Output: true

Constraints:
1 <= jug1Capacity, jug2Capacity, targetCapacity <= 106

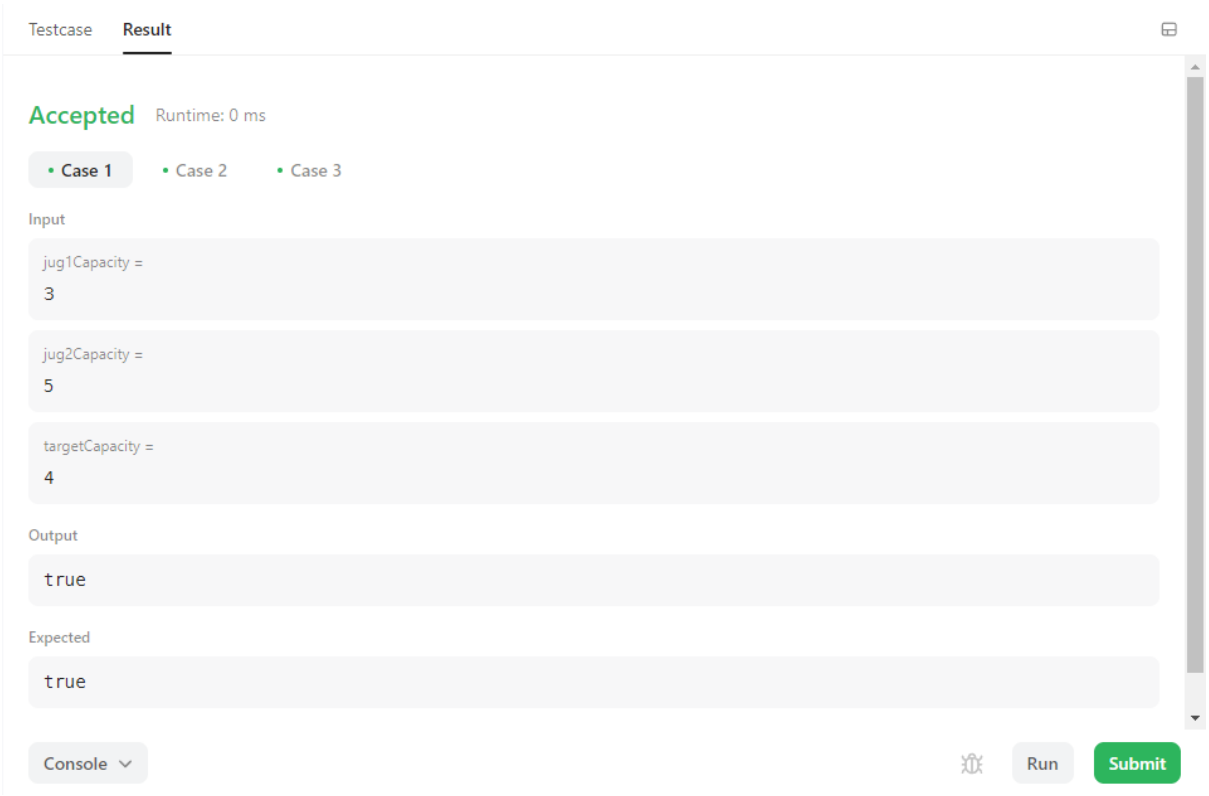
Code:

```
class Solution {
  public boolean canMeasureWater(int x, int y, int z) {
       if(x + y < z) return false;
      if( x == z || y == z || x + y == z ) return true;

    return z%GCD(x, y) == 0;
}

public int GCD(int a, int b){
    while(b != 0 ){
        int temp = b;
        b = a%b;
        a = temp;
    }
    return a;
}
}
```

Output:

Java • Auto

```java
class Solution {
  public boolean canMeasureWater(int x, int y, int z) {
        if(x + y < z) return false;
       if( x == z || y == z || x + y == z ) return true;

    return z%GCD(x, y) == 0;
}

public int GCD(int a, int b){
    while(b != 0 ){
        int temp = b;
        b = a%b;
        a = temp;
    }
    return a;
}
}
```



Submitted By – Avinash Jena