EXPERIMENT - 7

Student Name: Sandeep Kumar **UID: 20BCS4885**

Branch: CSE **Section/Group: 603/A**

Semester: 6th semester **Subject: Competitive Coding**

Aim: To demonstrate the concept of Divide and Conquer.

Objective:

a) Count and Say

b) Water and Jug Problem

Problem 1: Count and Say

Solution code:

```
class Solution {
public:
   string countAndSay(int n) {
      if(n == 1) {
         string str = "";
         str += '1';
         return str;
      }
      string ans = countAndSay(n-1);
      string str = "";
      for(int i=0; i<ans.size(); i++) {
         int count = 1;
         while(i != ans.size()-1 && ans[i] == ans[i+1]) {
            count++;
            i++;
         }
         str += (count + '0');
         str += ans[i];
      }
      return str;
   }
};
```

Approach:

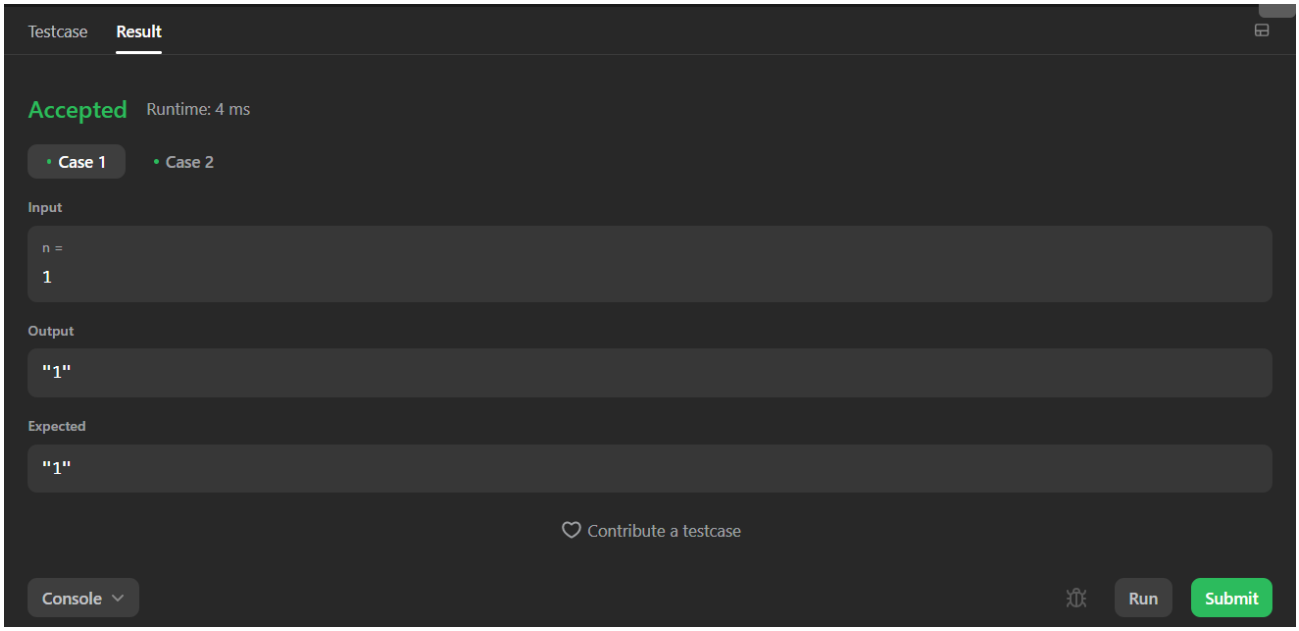
- The base case of the recursion is when n is equal to 1, in which case the function returns the string "1".
- In the recursive case, the function obtains the (n-1)th term of the sequence by calling itself recursively with n-1 as the input. It then constructs the nth term of the sequence by iterating through the (n-1)th term and counting the consecutive occurrences of each character.
- The algorithm counts the consecutive occurrences of each character by iterating through the string and checking if the current character is the same as the next character. If it is, it increments a count variable and moves to the next character. If it is not, it appends the count of the previous character to the new string, followed by the previous character.

Complexity:

Time Complexity: O(n)

Space Complexity: O(n)

Output:



Problem 2: Water and Jug Problem

Input Code:

```
class Solution {
public:
   bool canMeasureWater(int jug1Capacity, int jug2Capacity, int targetCapacity) {
      if(targetCapacity == 0)
      {
         return true;
      }
      if(targetCapacity > jug1Capacity + jug2Capacity)
      {
         return false;
      }
      else
      {
         int g = gcd(jug1Capacity,jug2Capacity);
         return targetCapacity % g == 0;
      }
   }
};
```

Approach:

- The function first checks the case where the target capacity is zero. If this is the case, it is always possible to measure zero water, so the function returns true.

- The next case to check is where the target capacity is greater than the sum of the capacities of both jugs. In this case, it is impossible to measure the target capacity using the given jugs, so the function returns false.

- If neither of the previous cases are true, then the function checks if the greatest common divisor (GCD) of the capacities of the two jugs divides the target capacity evenly. If it does, then it is possible to measure the target capacity using the given jugs, so the function returns true. Otherwise, it is impossible to measure the target capacity, so the function returns false.

Complexity:

Time Complexity: O(n)

Space Complexity: O(1)

Output:


Testcase
Result
Accepted
Runtime: 3 ms
Case 1
Case 2
Case 3
Input
jug1Capacity =
3
jug2Capacity =
5
targetCapacity =
4
Output
true
Console
Run
Submit