Count and Say

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

- `countAndSay(1) = "1"`
- `countAndSay(n)` is the run-length encoding of `countAndSay(n - 1)`.

Run-length encoding (RLE) is a string compression method that works by replacing consecutive identical characters (repeated 2 or more times) with the concatenation of the character and the number marking the count of the characters (length of the run). For example, to compress the string `"3322251"` we replace `"33"` with `"23"`, replace `"222"` with `"32"`, replace `"5"` with `"15"` and replace `"1"` with `"11"`. Thus the compressed string becomes `"23321511"`.

Given a positive integer `n`, return *the* `n^th` *element of the* ***count-and-say*** *sequence.*

Example 1:

> **Input:** `n = 4`
>
> **Output:** `"1211"`
>
> **Explanation:**
>
> ```
> countAndSay(1) = "1"
> countAndSay(2) = RLE of "1" = "11"
> countAndSay(3) = RLE of "11" = "21"
> countAndSay(4) = RLE of "21" = "1211"
> ```

Example 2:

> **Input:** `n = 1`
>
> **Output:** `"1"`
>
> **Explanation:**
>
> This is the base case.

Constraints:

- `1 <= n <= 30`

Follow up: Could you solve it iteratively?

vector [1]

c(1) = 1

c(2) = 11

c(3) = 21

c(4) =

```
class Solution {
public:
    string countAndSay(int n){
        return recur(n); }
    string recur(int n) {
    if(n==1) return "1";
     string x = recur(n-1);
     string ans = "";
      int cnt = 1;
      char prev = x[0];
      for(int i=1; i<x.length(); i++){
          if(x[i]==x[i-1]) cnt++;
          else {
        ans = ans + to_string(cnt);
        ans = ans + prev;
        cnt = 1; }
        prev = x[i]; }
```

ans = ans + to_string(cnt);

ans = ans + prev;

return ans; }};

C++ Auto

```cpp
class Solution {
public:
    string countAndSay(int n) {
        return recur(n);
    }
    string recur(int n){
        if(n==1)return "1";
        string x=recur(n-1);
        string ans="";
        int cnt=1;
        char prev=x[0];
        for(int i=1;i<x.length();i++){
            if(x[i]==x[i-1]){
                cnt++;
            }
            else{
                ans=ans+to_string(cnt);
                ans=ans+prev;
                cnt=1;
            }
                prev=x[i];
        }
         ans=ans+to_string(cnt);
                ans=ans+prev;
        return ans;
    }
};
```

Iterative :-

C++ Auto

```cpp
class Solution {
public:
    string countAndSay(int n) {
        if(n==1)return "1";
        string ans="1";
        for(int i=2;i<=n;i++){
            string cur="";
            char s=ans[0];
            int cnt=1;
            for(int i=1;i<ans.length();i++){
                if(ans[i]==ans[i-1]){
                cnt++;
                }
                else{
                    cur+=to_string(cnt);
                    cur+=s;
                    cnt=1;
                }
                s=ans[i];
            }
             cur+=to_string(cnt);
                cur+=s;
               ans=cur;
        }
        return ans;
    }
};
```