Solving the Problem II

Let's continue with the remaining steps in the checklist.

☐ Have You Covered All Possibilities?

We have (**at least**) two characters, (because otherwise, we would have gone into the base case). In any event, we only need to think about the first two characters.

- They **may** both match `c`. If that is the case, then we need to **ignore** the first character, and pass only a shortened string to the function.
- They **may not** both match `c`. If so, them we must **include** the first character in the returned string before passing a shortened string to our function.

This should **cover all possibilities.**

☐ Are the Subproblems Identical in Form?

Depending on whether both characters match `c` or not, we do a different action in each case; in one we include the first character, and in other we do not, but the **form** of the problem **is the same in both cases.**

☐ Are You a Believer?

Walk through a solution using the simplest recursive cases. If it works for those, then **it must work for all of them**.

1. Call `collapseSequences("vv", 'v')`
    - Both characters match `c`, so the first `v` is **ignored** and the function is **called again** with the string `"v"`.
    - On the second recursive call, the base case returns `"v"`
    - Thus, the sequence `"vv"` is compressed to `"v"`, **so it works**.
2. Now call `collapseSequences("va", 'v')`
    - The two characters **do not match**, so the first `'v'` is returned along with the value returned on the recursive call.
    - The second recursive call returns `"a"` from the base case.
    - Thus, the sequence `"va"` **is not compressed** (even though one of the characters matched the character `c`). **So, it works.**

Implement the algorithm described here and you'll see that all of the tests should pass. Now you can try a few problems on your own.