Solving the Problem I

Let's walk through the steps listed in the checklist. Normally, of course, you'll compress these steps together in an intuitive way. However, if you have difficulty with solving a recursive problem, going back and checking them individually is a good debugging tool.

☐ Check the Simple (Base) Cases

What is the simplest base case? In other words, what values for `s` require **no compression**? Well, obviously, if we don't have any characters, there can be no compression. Similarly, if we have only a single character, there can be no compression. The **simplest thing that can work, without recursion** is:

```
string collapseSequences(const string& s, char c)
{
    if (size() < 2) return s;  // base case
}
```

☐ Have You Solved the Base Cases Correctly?

Have we handled all of the base cases? The empty string returns `""`, and a single character returns that character. It doesn't matter if the character matches the parameter `c`, since a single-character cannot be a sequence. Those should be all of the base cases. If we have two charcters, we may have a sequence `"cc"` that requires compression.

☐ Does Decomposition Make it Simpler?

Does decomposition make it simpler? Or, in other words, how can we solve a simpler version of the problem? Or, how can we approach the base case? How? By **passing a smaller** string each time we call the function recursively.

This course content is offered under a CC Attribution Non-Commercial license. Content in this course can be considered under this license unless otherwise noted.