ately, without necessarily being able to easily decode it manually from the expression. Here is where **auto**&& forwarding references can be used effectively to decompose and document this expression while achieving the same result:

```
std::map<std::string, SpellingSuggestion> checkFileSpelling(
                                                    const std::string& filename)
{
    // Create a range over the contents of filename.
    auto&& openedFile = openFile(filename);

    // Split the file by whitespace.
    auto&& potentialWords = splitRegex(
        std::forward<decltype(openedFile)>(openedFile), "\\s+");

    // Filter out only words made from word-characters.
    auto&& words = filterRegex(
        std::forward<decltype(potentialWords)>(potentialWords), "\\w+");

    // Sort all words.
    auto&& sortedWords = sort(std::forward<decltype(words)>(words));

    // Skip adjacent duplicate words so as to create a sequence of unique words.
    auto&& uniqueWords = uniq(std::forward<decltype(sortedWords)>(sortedWords));

    // Get a SpellingSuggestion for every word.
    auto&& suggestions = transform(
        std::forward<decltype(uniqueWords)>(uniqueWords),
        [](const std::string&x) {
            return ~~std::tuple<std::string,SpellingSuggestion>(~~
                ~~x,checkSpelling(x));~~
        });

    // Filter out correctly spelled words, keeping only elements where the
    // second element of the tuple, which is a SpellingSuggestion, is not
    // correct.
    auto&& corrections = filter(
        std::forward<decltype(suggestions)>(suggestions),
        [](auto&& suggestion){ return !std::get<1>(suggestion).isCorrect(); });

    // Return a map made from these two-element tuples:
    return makeMap(std::forward<decltype(corrections)>(corrections));
}
```

Now each step of this complex expression is documented, each temporary has a name, but the net result of the lifetimes of each object is functionally the same. No new conversions have been introduced, and every object that was used as an *rvalue* in the original expression will still be used as an *rvalue* in this much longer and more descriptive implementation of the same functionality.