

Section Solution

Discussion Problem Solution 1: Publishing Stories

There's the one-pass approach that just appends characters from the template to the running story, unless that character is '{', in which case we extract everything up through the matching '}' and substitute it with a string from the data map.

```
string substituteTokens(string storyTemplate, Map<string>& data) {
    string story;
    for (int i = 0; i < storyTemplate.size(); i++) {
        if (storyTemplate[i] != '{') {
            story += storyTemplate[i];
        } else {
            int j = storyTemplate.find('}', i + 1);
            string token = storyTemplate.substr(i + 1, j - i - 1);
            story += data.get(token);
            i = j;
        }
    }
    return story;
}
```

Another approach is to iterate over the data map and drive the substitution that way. It's less efficient, but it's another way to think about the problem and is a perfectly acceptable answer for the purposes of a discussion section.

```
string substituteOneToken(string story, string token, string value) {
    int start = 0;
    while (true) {
        int found = story.find(token);
        if (found == string::npos) return story;
        story.replace(found, token.size(), value);
        start = found + value.size() + 1;
    }
}

string substituteTokens(string storyTemplate, Map<string>& data) {
    string story = storyTemplate;
    foreach (string token in data) {
        story = substituteOneToken(story,
                                   '{' + token + '}',
                                   data.get(token));
    }
    return story;
}
```

Discussion Problem Solution 2: Superwords

```

bool isSuperWord(string word, Lexicon& english) {
    for (int i = 1; i < word.size(); i++) {
        if (!english.containsWord(word.substr(0, i))) {
            return false;
        }
    }

    for (int i = 1; i < word.size(); i++) {
        if (!english.containsWord(word.substr(i))) {
            return false;
        }
    }

    return true;
}

Vector<string> collectLongestSuperwords(Lexicon& english) {
    int length = 0;
    Vector<string> longestSuperwords;

    foreach (string word in english) {
        if (isSuperWord(word, english) && word.size() >= length) {
            if (word.size() > length) {
                longestSuperwords.clear();
                length = word.size();
            }

            longestSuperwords.add(word);
        }
    }

    return longestSuperwords;
}

```

Lab Problem Solution 1: URL Parameter Map

The core of what needs to be written should be consistent with the implementation of **extractQueryMap** that I'm providing below. (The **explode** function is provided as part of the lab starter code.)

```

Map<string> extractQueryMap(string url) {
    Map<string> parameters;
    int index = url.find('?');
    if (index == string::npos) return parameters;

    string query = url.substr(index + 1);
    index = query.find('#');
    if (index != string::npos) {
        query = query.substr(0, index); // chop off hash
    }

    Vector<string> pairs = explode(query, '&');
    for (int i = 0; i < pairs.size(); i++) {
        Vector<string> components = explode(pairs[i], '=');
        string key = components[0];
    }
}

```

```

        string value = "true";
        if (components.size() > 1 && !components[1].empty()) {
            value = components[1];
        }
        parameters[key] = value;
    }

    return parameters;
}

```

Lab Problem Solution 2: Chain Reactions

Here's the core of my solution, with the code for the graphics and animation removed.

```

int computeScore(location init, Set<location>& landmines) {
    int score = 0;
    int second = 0;

    Set<location> notYetExploded = landmines;
    notYetExploded.remove(init);
    Set<location> currentlyExploding(LocationCompare);
    currentlyExploding.add(init);

    while (!currentlyExploding.isEmpty()) {
        score += 100 * currentlyExploding.size() * second * second;
        Set<location> soonExploding(LocationCompare);
        foreach (location explodingLocation in currentlyExploding) {
            foreach (location potentialLocation in notYetExploded) {
                if (isInRange(explodingLocation, potentialLocation)) {
                    soonExploding.add(potentialLocation);
                }
            }
        }

        notYetExploded.subtract(soonExploding);
        currentlyExploding = soonExploding;
        second++;
    }

    return score;
}

```