COMP220 — Exam 3 (100 points)

Fall 2017

For full credit, all implementations should make use of an ADT stack, queue, map, or set when/if appropriate. Vectors should only be used if those ADTs do not make sense.

1. Consider the design of a `std::queue` mutator procedure called `reverse` that reverses the contents of the queue.

   (a) Provide two gTest tests for the `reverse` mutator. One for the base case and one that is not a base case.

```
TEST(reverse, base) {
  std::queue<int> qu;
  qu.push(42); // empty queue is also acceptable
  std::queue<int> correct;
  correct.push(42);

  reverse(qu);
  EXPECT_EQ(qu, correct);
}

TEST(reverse, nonbase) {
  // bit of a trick for initialization
  // same as just pushing 1, 2, and 3 separately
  std::queue<int> qu({1,2,3});
  std::queue<int> correct({3,2,1});
  reverse(qu);
  EXPECT_EQ(qu, correct);
}
```

(b) Provide an implementation for the `reverse` mutator.

```
void reverse(std::queue<int>& qu) {
  std::stack<int> st;
  while (!qu.empty()) {
    st.push(qu.front());
    qu.pop();
  }
  while (!st.empty()) {
    qu.push(st.top());
    st.pop();
  }
}
```

2. Your boss has tasked you with doing a "frequency of length analysis" for a set of words/strings. That is, given a set of words, you want to know how many words in that set are of length 1, how many words are of length 2, how many are of length 3, etc.

   Your function should take in a set as the first parameter and another collection type as the second parameter. Your function should mutate this second parameter so that it contains the result.

   (a) Provide the documentation and declaration for a function that solves this problem.

```
/* Performs a length analysis on a set of words. That is,
 * counts the number of strings of length n in the set of
 * words, where n ranges over all the string lengths in
 * the set of words.
 * @param lexicon the set of words
 * @param counts the mapping from word
 *          length to frequency of occurrence in the lexicon.
 * @pre none
 * @post counts contains the actual mapping from
 *              word length to frequency.
 */
void freq(const set<string>& lexicon, map<int,int>& counts);
```

   (b) Write a single gTest test case for this problem. Try to cover as many of the basic logical cases of this problem as possible.

```
TEST(freq, all) {
  std::set<std::string> words{"a", "hat", "cat", "rat"};
  std::map<int,int> result;
  freq(words, result);
  EXPECT_EQ(result[1], 1); // single case
  EXPECT_EQ(result[2], 0); // empty case
  EXPECT_EQ(result[3], 3); // multiple case
}
```

(c) Provide a complete implementation for your function.

```
using namespace std;
void freq(const set<string>& lexicon, map<int,int>& counts) {
  for ( const string& word : lexicon ) {
    counts[word.length()]++;
  }
}
```