Section #2

Tyler Conklin and Kate Rydberg (Based on handouts by previous CS106B instructors and TAs, especially those of Ashley Taylor and Shreya Shankar.)

File I/O and ADTs

1. Input Stats (file I/O)

Write a function named `inputStats` that accepts a string parameter representing a file name, then opens/reads that file's contents and prints information to the console about the file's lines. Report the length of each line, the number of lines in the file, the length of the longest line, and the average characters per line. For example, if the input file carroll.txt contains the following data:

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun.
```

Then the call of `inputStats("carroll.txt");` should produce the following console output:

```
Line 1 has 30 chars
Line 2 has 41 chars
Line 3 has 31 chars
3 lines; longest = 41, average = 34
```

If the input file does not exist or is not readable, your function should print no output. If the file does exist, you may assume that the file contains at least 1 line of input. Your solution should read the file only once, not make multiple passes over the file data.

2. Collections Mystery (Stack and Queue)

Write the output produced by the following function when passed each of the following stacks and ints.

```
void collectionMystery10(Stack<int>& stack, int n) {
    Stack<int> stack2;
    Queue<int> queue;

    while (stack.size() > n) {
        queue.enqueue(stack.pop());
    }
    while (!stack.isEmpty()) {
        int element = stack.pop();
        stack2.push(element);
        if (element % 2 == 0) {
            queue.enqueue(element);
        }
    }
    while (!queue.isEmpty()) {
        stack.push(queue.dequeue());
    }
    while (!stack2.isEmpty()) {
        stack.push(stack2.pop());
    }

    cout << stack << endl;
}
```

1. $\{$**1, 2, 3, 4, 5, 6**$\}$**, n=3**

2. $\{$**67, 29, 115, 84, 33, 71, 90**$\}$**, n=5**

3. Reorder (Queue)

Write a function named `reorder` that accepts as a parameter a queue of integers that are already sorted by absolute value, and modifies it so that the integers are sorted normally. Only use a single stack as auxiliary storage. For example, if a queue variable named q stores the following elements:

`front {1, -2, 4, 5, -7, -9, -12, 28, -34} back`

Then the call of `reorder(q);` should modify it to store the following values:

`front {-34, -12, -9, -7, -2, 1, 4, 5, 28} back`

4. Check Balance (Stack)

Write a function named `checkBalance` that accepts a string of source code and uses a Stack to check whether the braces/parentheses are balanced. Every (or { must be closed by a } or) in the opposite order. Return the index at which an imbalance occurs, or -1 if the string is balanced. If any (or { are never closed, return the string's length.

Here are some example calls:

```
//    index   01234567890123456789012345678901234
checkBalance("if (a(4) > 9) { foo(a(2)); }")         // returns -1 because balanced
checkBalance("for (i=0;i&lt;a(3};i++) { foo{); )")   // returns 14 because } out of order
checkBalance("while (true) foo(); }{ ()")            // returns 20 because } doesn't match any {
checkBalance("if (x) {")                             // returns 8 because { is never closed
```

5. Big-O

Give a tight bound of the nearest runtime complexity class for the following code fragment in Big-Oh notation, in terms of the variable N.

```
Vector<int> v;
for (int i = 0; i < N; i++) {
    v.insert(0, i);
}
while (!v.isEmpty()) {
    v.remove(0);
}
cout << "done!" << endl;
```

6. Maps and Sets

Write a function named `friendList` that accepts a filename as a parameter, reads friend relationships from a file, and stores them into a compound collection that is returned. You should create a `Map` where each key is a persons name from the file, and the value associated with that key is a Set of all friends of that person. Friendships are bi-directional: if Tyler is friends with Kate, Kate is friends with Tyler.

The file contains one friend relationship per line, consisting of two names. The names are separated by a single space. You may assume that the file exists and is in a valid proper format.

If the file named buddies.txt looks like this:

```
Tyler Kate
Nick Tyler
```

Then the call of `friendList("buddies.txt")` should return a map with the following contents:

```
{"Tyler":{"Kate, Nick"}, "Kate":{"Tyler"}, "Nick":{"Tyler"}}
```

Constraints: You may open and read the file only once. Do not re-open it or rewind the stream. You may create one collection (Stack, Queue, Set, Map, etc.) or nested/compound structure as auxiliary storage. A nested structure, such as a set of vectors, counts as one collection.