Chris Gregg
CS106B

Section #2
April 18, 2018

Solution to Section #2

Based on handouts by various current and past CS106B/X instructors and TAs.

Sets

1. Twice

```
// solution
Set<int> twice(Vector<int> &v) {
    Map<int, int> counts;
    for (int i : v) {
        counts[i]++;
    }
    Set<int> twice;
    for (int i : counts) {
        if (counts[i] == 2) {
            twice += i;
        }
    }
    return twice;
}
```

```
// bonus
Set<int> twiceBonus(Vector<int> &v) {
    Set<int> once;
    Set<int> twice;
    Set<int> more;
    for (int i : v) {
      if (once.contains(i)) {
          once.remove(i);
          twice.add(i);
      } else if (twice.contains(i)) {
          twice.remove(i);
          more.add(i);
      } else if (!more.contains(i)) {
          once.add(i);
      }
    }
    return twice;
}
```

2. UnionSets

```
Set<int> unionSets(HashSet<Set<int>> &sets) {
     Set<int> all;
     for (Set<int> s : sets) {
          all += s;
     }
     return all;
}
```

Maps

1. Rarest

```
string rarest(Map<string, string> &map) {
    Map<string, int> counts;
    for (string key : map) {
        counts[map[key]]++;
    }
    string result = "";
    for (string s : counts) {
        if (result == "" || counts.get(s)< counts.get(result)) {
            result = s;  // Map sorts alphabetically for us!
        }
    }
    return result;
}
```

2. FriendList

```
Map<string, Vector<string>> friendList(string filename) {
     ifstream infile;
     openFile(infile, filename);
     Map<string, Vector<string>> friends;
     string s1, s2;
     while(infile >> s1 >> s2) {
          friends[s1] += s2;
          friends[s2] += s1;
     }
     return friends;
}
```

3. Reverse Map

```
Map<string, int> reverseMap(Map<int, string> &map) {
     Map<string, int> rev;
     for (int i : map) {
          rev[map[i]] = i;
     }
     return rev;
}
```

Recursion

1. Mystery Trace

Stacks: Output:

```
mystery1(4, 1)                    4
mystery1(8, 2)                    16, 8, 16
mystery1(3, 4)                    12, 9, 6, 3, 6, 9, 12
```

2. Sum of Squares

```
int sumOfSquares(int n) {
    if (n == 1) {
        return 1;
    } else {
        return n * n + sumOfSquares(n - 1);
    }
}
```

3. Reverse String

```
string reverseString(string s) {
    if (s.length() < 2) {
        return s;
    } else {
        return reverseString(s.substr(1)) + s[0];
    }
}
```

4. Star String

```
string starString(int n) {
    if (n == 0) {
        return "*";
    } else {
        string s = starString(n - 1);
        return s + s;
    }
}
```

5. Is Subsequence

```
bool isSubsequence(string big, string small) {
    if (small == "") {
        return true;
    } else if (big == "") {
        return false;
    } else {
        if (big[0] == small[0]) {
            string bigSubstr = big.substr(1);
            string smallSubstr = small.substr(1);
            return isSubsequence(bigSubstr, smallSubstr);
        } else {
            return isSubsequence(big.substr(1), small);
        }
    }
}
```

6. Double Stack

```
void doubleStack(Stack<int> &s) {
    if (!s.isEmpty()) {
        int n = s.pop();
        doubleStack(s);
        s.push(n);
        s.push(n);
    }
}
```

7. Zig Zag

```
void zigzag(int n) {
    if (n == 1) {
        cout << "*";
    } else if (n == 2) {
        cout << "**";
    } else {
        cout << "<";
        zigzag(n - 2);
        cout << ">";
    }
}
```

8. Directory Crawl

```
// Prints information about this file,
// and (if it is a directory) any files inside it.
void crawl(string filename, string indent) {
    cout << indent << getTail(filename) << endl;
    if (isDirectory(filename)) {
        // recursive case; print contained files/dirs
        Vector<string> filelist;
        listDirectory(filename, filelist);
        for (string subfile : filelist) {
            crawl(filename + "/" + subfile, indent + "    ");
        }
    }
}
```