CS106B Chris Gregg
Summer 2017

Section Handout #2 Solutions

If you have any questions about the solutions to the problems in this handout, feel free to reach out to your section leader, Jason, or Chris for more information.

1. Twice

```
// solution 1: with additional structures
Set<int> twice(Vector<int> &v) {
  Map<int, int> counts;
  for (int n : v) {
    counts[n]++;
  }

  Set<int> twice;
  for (int k : counts) {
    if (counts[k] == 2) {
      twice += k;
    }
  }
  return twice;
}
```

```
// solution 2: without auxiliary structures
Set<int> twice(Vector<int> &v) {
  Set<int> one, two, more;
  for (int n : v) {
    if (one.contains(n)) {
      one.remove(n);
      two.add(n);
    } else if (two.contains(n)) {
      two.remove(n);
      more.add(n);
    } else if (!more.contains(n)) {
      one.add(n);
    }
  }
  return two;
}
```

2. UnionSets

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

3. Reorder

```
void reorder(Queue<int> &q) {
  Stack<int> s;
  int size = q.size();
  for (int i = 0; i < size; i++) { // separate positive and negative numbers
    int n = q.dequeue();
    if (n < 0) {
      s.push(n);
    } else {
      q.enqueue(n);
    }
  }

  size = q.size();                 // enqueue negative numbers in reverse order
  while (!s.isEmpty()) {
      q.enqueue(s.pop());
  }
```

```
  for (int i = 0; i < size; i++) { // move positive numbers to end
      q.enqueue(q.dequeue());
  }
}
```

4. CheckBalance

```
int checkBalance(string code) {
    Stack<char> parens;
    for (int i = 0; i < (int) code.length(); i++) {
        char c = code[i];
        if (c == '(' || c == '{') {
            parens.push(c);
        } else if (c == ')' || c == '}') {
            if (parens.isEmpty()) {
                return i;
            }
            char top = parens.pop();
            if ((top == '(' && c != ')') || (top == '{' && c != '}')) {
                return i;
            }
        }
    }
    if (parens.isEmpty()) {
        return -1;            // balanced
    } else {
        return code.length();
    }
}
```

5. Friend List

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

6. Reverse

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

7. CrazyCaps

```
void crazyCaps(string& s) {
    for (int i = 0; i < s.length(); i++) {
        if (i % 2 == 0) {
            s[i] = tolower(s[i]);
        } else {
            s[i] = toupper(s[i]);
        }
    }
}
```

8. SwapPairs

```
void swapPairs(string& s) {
    for (int i = 0; i < s.length() - 1; i += 2) {
        char temp = s[i];
        s[i] = s[i + 1];
        s[i + 1] = temp;
    }
}
```