Chris Gregg
CS106B

Section #3
April 25, 2018

Solution to Section #3

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

1. Make Change

```
void makeChange(int amount, Vector<int> &coins) {
    Vector<int> chosen;
    makeChangeHelper(amount, coins, chosen);
}

void makeChangeHelper(int amount, Vector<int> &coins, Vector<int> &chosen) {
    if (coins.isEmpty()) {
        if (amount == 0) {
            cout << chosen << endl;
        }
    } else {
        // Choose a coin, and explore all quantities of this coin
        int coin = coins[0];
        coins.remove(0);
        for (int i = 0; i <= (amount / coin); i++) {
            chosen += i;
            makeChangeHelper(amount - (i * coin), coins, chosen);
            chosen.remove(chosen.size() - 1);
        }
        // Un-choose a coin
        coins.insert(0, coin);
    }
}
```

2. Print Squares

```
void printSquares(int n) {
    Set<int> chosen;
    printSquaresHelper(n, 1, chosen);
}

void printSquaresHelper(int n, int min, Set<int> &chosen) {
    if (n < 0) {
        return;
    } else if (n == 0) {
        cout << chosen << endl;
    } else {
        int max = (int) sqrt(n); // valid choices go up to sqrt(n)
        for (int i = min; i <= max; i++) {
            chosen.add(i);      // choose
            printSquaresHelper(n - (i * i), i + 1, chosen);     // explore
            chosen.remove(i);   // un-choose
        }
    }
}
```

3. Longest Common Subsequence

```
string longestCommonSubsequence(string s1, string s2) {
    if (s1.length() == 0 || s2.length() == 0) {
        return "";
    } else if (s1[0] == s2[0]) {
        return s1[0] + longestCommonSubsequence(s1.substr(1), s2.substr(1));
    } else {
        string choice1 = longestCommonSubsequence(s1, s2.substr(1));
        string choice2 = longestCommonSubsequence(s1.substr(1), s2);
        if (choice1.length() >= choice2.length()) {
            return choice1;
        } else {
            return choice2;
        }
    }
}
```

4. Ways to Climb

```
void waysToClimb(int stairs) {
    Stack<int> chosen;
    waysToClimbHelper(stairs, chosen);
}

void waysToClimbHelper(int stairs, Stack<int> &chosen) {
    if (stairs < 0) {
        return;
    } else if (stairs == 0) {
        cout << chosen << endl;
    } else {
        chosen.push(1);     // choose 1
        waysToClimbHelper(stairs - 1, chosen);      // explore
        chosen.pop();       // un-choose 1

        chosen.push(2);     // choose 2
        waysToClimbHelper(stairs - 2, chosen);      // explore
        chosen.pop();       // un-choose 2
    }
}
```

5. Twiddle

```
void listTwiddles(string str, const Lexicon &lex) {
    string prefix = "";
    listTwiddlesHelper(prefix, str, /* index */ 0, lex);
}

void listTwiddlesHelper(string prefix, string str, int index,
    const Lexicon &lex) {

    if (!lex.containsPrefix(prefix)) {
        return; // optimization; not strictly necessary but good to do
    }
    if (index >= (int)str.size()) {
        if (lex.contains(prefix)) {
            cout << prefix << endl;
        }
    } else {
        for (char ch = str[index] - 2; ch <= str[index] + 2; ch++) {
            if (isalpha(ch)) {
                listTwiddlesHelper(prefix + ch, str, index + 1, lex);
            }
        }
    }
}
```

6. Password Cracking

```
string findPassword(int maxLength) {
    if (maxLength < 0) {
        throw maxLength;
    } else if (maxLength == 0) {
        return "";
    }
    return findPasswordHelper("", maxLength);
}

string findPasswordHelper(string soFar, int maxLength) {
    if (login(soFar)) {
        return soFar;
    }

    if ((int)soFar.size() == maxLength) {
        return "";
    }

    for (char c = 'a'; c <= 'z'; c++) {
        string password = findPasswordHelper(soFar + c, maxLength);
        if (password != "") {
            return password;
        }

        // Also check uppercase
        char upperC = toupper(c);
        password = findPasswordHelper(soFar + upperC, maxLength);
        if (password != "") {
            return password;
        }
    }

    return "";
}
```