Section #4: Solutions

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

Recursion and Recursive Backtracking: Solutions

1. Can Make Sum

```
bool canMakeSumHelper(Vector<int>& v, int target, int cur, int index) {
    if (index >= v.size()) {
        return cur == target;
    }
    return canMakeSumHelper(v, target, cur + v[index], index + 1) ||
           canMakeSumHelper(v, target, cur, index + 1);
}

bool canMakeSum(Vector<int>& v, int target) {
    return canMakeSumHelper(v, target, 0, 0);
}
```

2. Edit Distance

```
int editDistance(const string& s1, const string& s2) {
    if (s1 == "") {
        return s2.length();
    } else if (s2 == "") {
        return s1.length();
    }
    int add = 1 + editDistance(s1, s2.substr(1, s2.length()));
    int del = 1 + editDistance(s1.substr(1, s1.length()), s2);
    int sub = editDistance(s1.substr(1, s1.length()), s2.substr(1, s2.length()));
    if (s1[0] != s2[0]) {
        sub++;
    }
    return min(add, min(del, sub));
}
```

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. Print Squares

```
void printSquaresHelper(int n, int min, Set<int>& chosen) {
    if (n == 0) {
        display(chosen); // base case: sum has reached n
    } else {
        int max = sqrt(n); // valid choices go up to sqrt(n)
        for (int i = min; i <= max; i++) {
            chosen.add(i);
            printSquaresHelper(n - (i * i), i + 1, chosen);
            chosen.remove(i); // backtrack
        }
    }
}

void printSquares(int n) {
    if (n < 0) {
        throw n;
    }
    Set<int> chosen;
    printSquaresHelper(n, 1, chosen);
}
```

5. Crack

```
string crackHelper(string soFar, int maxLength) {
    if (login(soFar)) {
        return soFar;
    }
    if (soFar.size() == maxLength) {
        return "";
    }
    for (char c = 'a'; c <= 'z'; c++) {
        string password = crackHelper (soFar + c, maxLength);
        if (password != "") {
            return password;
        }
        // Also check uppercase
        char upperC = toupper(c);
        password = crackHelper (soFar + upperC, maxLength);
        if (password != "") {
            return password;
        }
    }
    return "";
}

string crack(int maxLength) {
    if (maxLength < 0) {
        throw maxLength;
    }
    return crackHelper("", maxLength);
}
```