CS106B
Spring 2016 Cynthia Lee

Section 3 (Week 4) - SOLUTION

1. stutter.

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

2. starString.

```
string starString(int n) {
    if (n < 0) {
        throw "Invalid input.";
    } else if (n == 0) {
        return "*";
    } else {
        // Inefficient solution:
        // return starString(n – 1) + starString(n – 1);
        // That requires an exponential number of recursive calls
        // because every function call makes two more.
        string stars = starString(n – 1);
        return stars + stars;
    }
}
```

3. writeChars.

```
void writeChars(int n) {
    if (n < 1) {
        throw "Invalid input.";
    } else if (n == 1) {
        cout << "*";
    } else if (n == 2) {
        cout << "**";
    } else {
        cout << "<";
        writeChars(n – 2);
        cout << ">";
    }
}
```

4. isMeasurable.

```
bool isMeasurable(int target, Vector<int>& weights) {
    if (weights.isEmpty()) {
        return target == 0;
    }
    int first = weights[0];
    Vector<int> rest = weights;
    rest.remove(0);
    return isMeasurable(target, rest)
        || isMeasurable(target – first, rest)
        || isMeasurable(target + first, rest);
}
```

5. waysToClimb.

```
int waysToClimb(int steps) {
    if (steps <= 0) {
        throw "Invalid input.";
    } else if (steps <= 2) {
        return steps; // 1 way to climb 1 step and
                      // 2 ways to climb 2 steps
    } else {
        return waysToClimb(steps – 1) +
               waysToClimb(steps – 2);
    }
}
```

6. isSubsequence.

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

7. Debugging Recursion.

The middle index is repeated in both sub-ranges, so when the recursion gets down to a range of length 2, the recursive call doesn't actually get any smaller, so it loops infinitely. Fix: replace the second recursive call with `recursiveMax(v, middle + 1, right)`.