Section #3

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

Recursion

1. Recursion Mystery A

```
int mysteryA(int x, int y) {
    if (x < y) {
        return x;
    } else {
        return mysteryA(x - y, y);
    }
}
```

For each of the following calls to `mysteryA`, indicate what is returned:

```
mysteryA(6, 13)
mysteryA(14, 10)
mysteryA(37, 12)
```

2. Recursion Mystery B

```
void mysteryB(int x, int y) {
    if (y == 1) {
        cout << x;
    } else {
        cout << (x * y) << , ;
        mysteryB(x, y - 1);
        cout << ,  << (x * y);
    }
}
```

For each of the following calls to `mysteryB`, indicate what is printed:

```
mysteryB(4, 1)
mysteryB(8, 2)
mysteryB(3, 4)
```

3. Star String

Write a recursive function named `starString` that accepts an integer parameter `n` and returns a string of stars (asterisks) `2n` long (i.e., 2 to the nth power).

For example,

```
starString(0)     returns *
starString(1)     returns **
starString(2)     returns ****
starString(3)     returns ********
```

Do not use loops or auxiliary data structures; solve the problem recursively. You may assume that the value passed is non-negative.

4. Sum of Squares

Write a recursive function named `sumOfSquares` that accepts an integer parameter `n` and returns the sum of squares from 1 to n. For example, the call of `sumOfSquares(3)` should return 12 + 22 + 32 = 14. If your function is passed 0, return 0. If passed a negative number, your function should throw an `int` as an exception.

5. Stutter Stack

Write a recursive function named `stutterStack` that takes a reference to a stack of ints and replaces each integer with two copies of that integer.

For example, if `s` stores {`1, 2, 3`}, then `stutterStack(s)` changes it to {`1, 1, 2, 2, 3, 3`}.

6. Greatest Common Divisor

The greatest common divisor (often abbreviated to GCD) of two nonnegative integers is the largest integer that divides evenly into both. In the third century BCE, the Greek mathematician Euclid discovered that the greatest common divisor of x and y can be computed as follows:

- If x is evenly divisible by y, then y is the greatest common divisor.
- Otherwise, the greatest common divisor of x and y is always equal to the greatest common divisor of y and the remainder of x divided by y.

Use Euclids insight to write a recursive function `gcd` that takes two integers, `x` and `y`, and computes the greatest common divisor of x and y. If either x or y are 0 or negative, throw an error.

7. Digit Sum

Write a recursive function `digitSum` that takes an integer and returns the sum of its digits. For example, calling `digitSum(1729)` should return `1 + 7 + 2 + 9`, which is `19`, and calling `digitSum(-1729)` should return `-19`.

The recursive implementation of `digitSum` depends on the fact that it is very easy to break an integer down into two components using division by 10.

For example, given the integer `1729`, you can divide it into two pieces by dividing by 10: 172 and 9, each of which are strictly smaller than 1729 and thus represent a simpler case.

8. Zig Zag

Write a recursive function named `zigzag` that prints `n` characters as follows. The middle character (or middle two characters if `n` is even) is an asterisk (*). All characters before the asterisks are '¡'. All characters after are '¿'. You may assume `n` is not zero or negative. (You do not need to worry about printing an `endl` at the end.)

```
zigzag(1)     *
zigzag(4)     <**>
zigzag(9)     <<<<*>>>>
```

9. Is Subsequence

Write a recursive function named `isSubsequence` that accepts two string parameters, and returns `true` if the second string is a subsequence of the first string. A string is a subsequence of another if it contains the same letters in the same order, but not necessary consecutively. You can assume both strings are already lowercased.

```
isSubsequence("computer", "core")         false
isSubsequence("computer", "cope")         true
isSubsequence("computer", "computer")     true
```