Week 3 Section

This week's section handout has practice with some initial recursion problems.

---

1. RecursionMysteryComma - Recursion Trace

For each call to the following recursive function, write the output that would be produced, as it would appear on the console.

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

Output:

`recursionMysteryComma(4, 1);` _____

`recursionMysteryComma(4, 2);` _____

`recursionMysteryComma(8, 2);` _____

`recursionMysteryComma(4, 3);` _____

`recursionMysteryComma(3, 4);` _____

---

2. sumOfSquares - Recursion

Write a recursive function named **`sumOfSquares`** that takes an int *n* and returns the sum of squares from 1 to *n*. For example, `sumOfSquares(3)` should return $1^2 + 2^2 + 3^2 = 14$. (You can assume that *n* is ≥ 1).

---

3. Zigzag - Recursion

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 `'>'`. Throw an error if *n* is not positive. (You do not need to worry about printing an endl at the end.)

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

Thanks to CS106B and X instructors and TAs for contributing problems on this handout.

Week 3 Section

4. StutterStack - Recursion

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}`.

---

5. IsSubsequence - Recursion

Write a recursive function named **`isSubsequence`** that takes two strings 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 necessarily consecutively. You can assume both strings are already lower-cased.

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

---

6. reverseLines - Recursion, File I/O

Write a recursive function named **`reverseLines`** that accepts as its parameter a reference to a file input stream (`ifstream`) and prints the lines of that file in reverse order. For example, if an input file named `poem.txt` contains the following text:

```
Roses are red,
Violets are blue.
All my base
Are belong to you.
```

Then the call of `reverseLines("poem.txt");` should produce the following console output:

```
Are belong to you.
All my base
Violets are blue.
Roses are red,
```

You may assume that the input file exists and is readable.

Thanks to CS106B and X instructors and TAs for contributing problems on this handout.