Homework 3

1.

```
/**
 * Smooths a given {@code Sequence<Integer>}.
 *
 * @param s1
 *            the sequence to smooth
 * @param s2
 *            the resulting sequence
 *
 * @requires |s1| >= 1
 * @ensures <pre>
 * |result| = |s1| - 1  and
 *  for all i, j: integer, a, b: string of integer
 *      where (s1 = a * <i> * <j> * b)
 *    (there exists c, d: string of integer
 *       (|c| = |a|  and
 *        result = c * <(i+j)/2> * d))
 * </pre>
 * @returns result
 */
public static Sequence<Integer> smooth(Sequence<Integer> s1, Sequence<Integer>
s2) {...}
```

Iterative Implementation:

```
      Int i = 0;
      Int j = 0;
      Sequence<Integer> result = <>;

      if (s1.length() > 1) {

            while (idx + 2 <= s1.length()) {

                  // pull values from s1
                  i = s1.remove(idx);
                  j = s1.remove(idx);

                  // take avg of each group of 2 nums
                  avg = (int) ((i / 2.0) + (j / 2.0));

                  // put each avg in result
                  result.add(result.length(), avg);

                  // return values to s1
                  s1.add(idx, i);
                  s1.add(idx + 1, j);

                  // iterate
                  idx++;
            }
      }
```

Recursive Implementation:

```
Int i = 0;
Int j = 0;
Int avg = 0;
Sequence<Integer> result = <>;

if (s1.length() > 1) {

      while (idx + 2 <= s1.length()) {

            // pull values from s1
            i = s1.remove(idx);
            j = s1.remove(idx);

            // take avg of each group of 2 nums
            avg = (int) ((i / 2.0) + (j / 2.0));

            // put each avg in result
            Result = smooth(s1, s2);

            Result.add(0, avg);

            // return values to s1
            s1.add(idx, i);
            s1.add(idx + 1, j);

            // iterate
            idx++;
      }
}
```

Return result;

}