1. It fails the testThrowsIllegalArgumentException since in line 23 it says “if (n <= 0)” rather than “if (n < 0)” which would only cause the exception to be thrown for only negatives. To fix this you had to make it “if (n < 0)”.
2. The testBaseCase failed because 0 was already throwing an exception in line 24 "throw new IllegalArgumentException(n + " is negative");” rather than returning 0 under line 25 “else if (n <= 2)”. To fix this you would have to have the “if (n < 0)” and also add “if(n == 0) {return 0;}” under “else if (n <= 2)” and also add “else {return 1;}” so that it doesn’t mess up the 1.
3. The testInductiveCase failed because the fibonacci sequence for the code wasn’t even right, it should’ve been “return getFibTerm(n - 2) + getFibTerm(n - 1); “ instead of “return getFibTerm(n + 1) - getFibTerm(n - 2);” this way it adds the two previous numbers together rather than subtract a rather number.
4. The original code failed the test because it returns only ints rather than longs and ints are only a certain amount big and the fib sequence of 60 just had too big of a number so all you had to do to fix this is turn “public int getFibTerm(int n)” into “public long getFibTerm(int n)” in order to return such a big number.
5. The original code is slow because it is calling some fibs multiple times, for example just running fib (5) calculates fib(2) multiple times having to recalculate what fib(2) is over and over again. This can be solved by using a map. I add “import java.util.HashMap; import java.util.Map;” so that I’m able to use maps then I create a map for memory called “private Map<Integer, Long> memo = new HashMap<>();” which returns the long result for the specific imputed integer. For example an input of 8 would return the 8th long in the fib sequence which is 21. I then made it so that the function checks if the map already has the input by doing “if (memo.containsKey(n)) { return memo.get(n); }” which returns the long if it has already been calculated. I then add "memo.put(n, result);” so that it adds the newest n result so that it would be able to check the memory to see what long would be for the next n input.