|  |
| --- |
| Ricky Mutschlechner  9/6/2012 |

|  |
| --- |
| CSIS 235 Lab 1 Report |
|  |
|  |

1. When you add +1 to Integer.MAX\_VALUE (2147483647), you actually get Integer.MIN\_VALUE (-2147483648). I assume this happens because you are basically “resetting” the int from the highest value possible to the lowest value possible due to how memory works.
2. When you add +1 to Float.MAX\_VALUE(3.4028235E38), you get Float.POSITIVE\_INFINITY. I believe this is due to the fact that Float.POSITIVE\_INFINITY is next in the memory, therefore when you overflow from Float.MAX\_VALUE, you get “put” into Float.POSITIVE\_INFINITY.
3. It is okay to add these numbers and not crash, because Java seems to ignore the number, I assume because it is too small. If you actually do add a number like 1, 10, or 100 it simply ignores it or so it seems.
4. You would have to add another instance of infinity in order to crash the program, or add something that is not an actual number, such as a character or string.
5. When you decrease the smallest float number by any other number, it seems to reset the float back to zero, and then do the decreasing. I subtracted 1.001 from Float.MIN\_VALUE and my program printed -1.001.
6. When I change the value of b in bad\_subtraction\_method, it seems that changing it to 0.7, 0.8, and 0.9 causes it not to work properly, for example: changing b to 0.7 gives us: a is 1.0 and b is 0.7

a - b is 0.30000000000000004