Jacob Knott
11398813
CS 422
HW14

1) When sorting, if the number above is less than the number below (needs to be moved) then the algorithm will move the number correctly. However, it will at most do this once. For example, if you have 3, 2, 1. Then, 3 is less than 2 so they will switch and the array will be 2, 3, 1, then as it continues it will see 1 is less than 3 so it will switch and become 2, 1, 3. Clearly that is not sorted. The simple solution is to decrease j by one every time we make a shift inside of the while loop. This will update the while loops condition and result in a sorted array.

2) The problem is that Type.GetMethod(string) method searches for the public method with the given name. By not giving the function an access modifier it will default to internal. This is a problem because then the GetMethod will not be able to find it. The simple fix is to make 'Function' public.

3)

Lossless
- 4.25 can be fully represented in binary, 4.25 = 100.01

Lossless
- Since 4.25 is lossless, then we can store that number in binary, then we can copy it's full value to another variable.

Lossy
- 1.4 cannot be fully represented in binary. Therefore, the computer will try to represent it the best it can and fill up all of the 52 bits it as for the fraction part of the float.

Lossless
- Even though 1.4 is lossy, when we assign it to the double d2, then all the bits are copied over and f2 is equal to d2, even though internally, neither are exactly 1.4.

4) Since the int is not a reference type then when we make an object (that is a reference type) o, then o will be a reference to an object that has the value of the int we assigned it to. Basically, o is a reference but, not a reference to the int object. It is a reference to the value of the int but num and o are different objects. Therefore, the output is what we would expect it to be. When we do num++, we are modifying num and not o, since o is not a reference to num.

5) The problem is that each of methods should have a space after them. The specs say that the method is followed by a space. So, a method with the name ‘GETjdhdfefad’ would return as a valid http method.

6) Given the assumption that the integers could be in the range of [int.MinValue, int.MaxValue] then none of the operations are lossless. If both a and b are int.MaxValue, then a + b is lossy, since, depending of compiler, will just wrap and loose your number. If both a and b are int.MinValue then a - b will also be lossy. Again, if both a and b are int.MaxValue, then a*b has a similar issue as a + b. Finally, integer division is inherently lossy since there are integers a and b such that a/b is not an integer, and thus, lossy.

7) Method 1 is the best, because it will simply see if the file exists and return true or false. Method 2 is bad because it will try to open the file, and when it cannot find it, it will throw an IO, file not found exception. Also, if the file did exists then method 2 will end up opening the file. This is bad because a) if the file is already open we may get an exception even though the file exists, and b) we are not closing the file, we open it and then just return. Method 3 is also bad since it will also throw an IO file not found exception if the file is not found.

8) One major issue with anonymous types is that their fields are read only. So you cannot change the value of an item, or change what the next node is in the list, once initialized we cannot modify the list.

9) The problem here is that incrementing the variable s_sharedValue is not an atomic action, it is possible that both threads are in the ‘middle’ of incrementing the value at the same time and thus we loose one of those additions. A way that we can make this addition atomic without using the lock statement is to use Interlocked.Increment(ref s_sharedValue) instead of s_sharedValue++. By using the Interlocked.Increment the addition becomes atomic and will be thread safe.

10) The string I chose was “\b”. I choose this because it is the encoded char for backspace. By doing this, when the doc.ToString() is executed it will try to interpret this char as a backspace and will cause the string to be messed up and throw an exception.