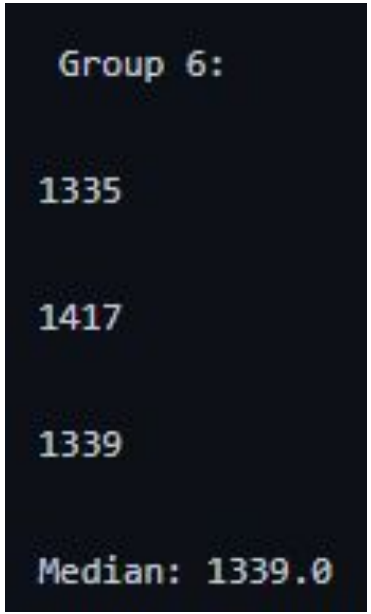
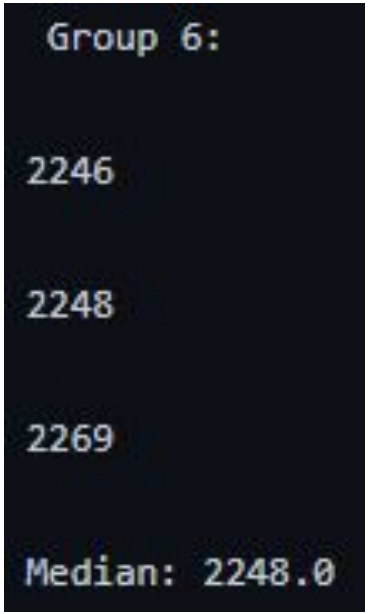
Group 6 Sorting Competition

Members: Nicholas Gilbertson, Dylan Cramer

Results/Correctness Issues






```
group 12 took place  1. The sum of places is 2, the sum of medians is 99.0

group 11 took place  2. The sum of places is 4, the sum of medians is 162.0

group 8 took place  3. The sum of places is 6, the sum of medians is 309.0

group 10 took place  4. The sum of places is 8, the sum of medians is 349.0

group 9 took place  5. The sum of places is 10, the sum of medians is 1130.0

group 6 took place  6. The sum of places is 12, the sum of medians is 3587.0

group 1 took place  7. The sum of places is 14, the sum of medians is 6917.0

group 5 took place  8. The sum of places is 16, the sum of medians is 7125.0

group 7 took place  9. The sum of places is 19, the sum of medians is 8407.0

group 3 took place  10. The sum of places is 19, the sum of medians is 8663.0

group 2 took place  11. The sum of places is 22, the sum of medians is 10562.0

group 4 took place  12. The sum of places is 24, the sum of medians is 16648.0

group 0 took place  13. The sum of places is 26, the sum of medians is 17424.0
```

Changes Made to Group 0

- Added new class with Object type obj
- obj has fields: num1s, repeatLength, name, and binRep
- The goal of this object is to store the data relating to an entry in the array so that it doesn't have to be re-calculated during each comparison
- Data is stored in an obj array (obj[]) and writes over the initial array at the end after sorting using each obj's name attribute.

```java
public int compare(obj n1, obj n2) {
    if(n1.getBinRep().equals("-1")){
        n1.setNum1s();
    }
    if(n2.getBinRep().equals("-1")){
        n2.setNum1s();
    }
```

```java
public class obj {
    private int num1s;
    private int repeatLength;
    private int name;
    private String binRep;

    public obj(int name) {
        this.name = name;
        this.binRep = "-1";
        this.repeatLength = -1;
    }

    public void setNum1s() {
        this.binRep = Integer.toBinaryString(this.name);
        this.num1s = Helperó.numBinaryOnes(this.binRep);
    }

    public int getNum1s() { return num1s; }

    public int getRepeatLength() { return repeatLength; }

    public void setRepeatLength(int repeatLength) {

        this.repeatLength = Helperó.lengthLongestRepeatedSubstring(this.binRep);
    }

    public String getBinRep() { return binRep; }

    public int getName() { return name; }

    public void setName(int name) { this.name = name; }
```

Running Time

- Running time is n(m^4) in worst case, because `lengthLongestRepeatedSubstring` has efficiency of m^4 and it would be used n times in the worst case. (m is length of binRep)
- In the original implementation `lengthLongestRepeatedSubstring` was used in the comparator many times, the object oriented implementation mitigates it's bad efficiency by only needing to run it once per object

```java
public static int lengthLongestRepeatedSubstring(String binary) {
    int length = 0;
    // iterate over possible lengths
    // the longest length is length/2 (rounded down) since they are non-overlapping
    for (int n = 1; n <= Math.floor(binary.length()/2.0); ++n) {
        //System.out.println("n = " + n);
        boolean found = false;
        // first index (the first index of the first copy):
        for (int i = 0; i < binary.length() - 2*n + 1; ++i) {
            // second index (substrings are non-overlapping):
            for (int j = i + n; j < binary.length() - n + 1; ++j) {
                // iterating over the substring length:
                int k = 0; // need the index after the loop to see if it finished
                for (; k < n; k++) {
                    //System.out.println("i = " + i + ", j = " + j + ", k = " + k);
                    if (binary.charAt(i + k) != binary.charAt(j + k)) {
                        break;
                    }
                }
                //System.out.println("k = " + k + ",n = " + n);
                if (k == n) {
                    found = true;
                }
            }
        }
        //System.out.println(found);
        if (found) {
            length++;
        } else {
            return length;
        }
    }

    return length;
}
```