CS305j Spring 2010

Midterm 2 Key, Suggested Solutions, and Grading Criteria

Abbreviations:

NAP - no answer provided
ECF - error carried forward
OBOE - off by one error
BOD - benefit of the doubt
GCE - misunderstood question. Answer is way off base.
NN - Not Necessary. The code was not required. Usually no points off for this.
AIOBE - Array Index out of Bounds Exception may occur
GACK - Inelegant code or code very hard to understand even though it works. (Generally no points off for this.)

1A. 4 points. Moore's law describes a trend over the past 40 years that the density of transistors on integrated circuits doubles about every two years. This has led to an exponential growth in computing power.

1B. 4 points. Computation and programming has aided in the search for new medicinal drugs by allowing the modeling and simulation of potential new drugs. This helps reduce the trial and error and allows for more ideas to be tested quickly.

2. (1 point per number.)
Line 1: 13 4
Line 2: 5 12
Line 3: 5 7
Line 4: 6 9
Line 5: 6 9
Line 6: 0 5
Line 7: 12 19
Line 8: 1 0

3.
```
public static boolean areParallel(int x1, int y1, int x2,
      int y2, int x3, int y3, int x4, int y4) {

      boolean result = false;
      if(x1 == x2 || x3 == x4)
            result = x1 == x2 && x3 == x4;
      else{
            double slope1 = (1.0 * y1 - y2) / (1.0 * x1 - x2);
            double slope2 = (1.0 * y3 - y4) / (1.0 * x3 - x4);
            result = slope1 == slope2;
      }
      return result;
}
```

calculate slope correctly: 10 points
ensure division is floating point (double) division: 2 points
handle case when slope is infinite. (divide by 0): 3 points
return correct boolean result: 3 points

Common problems were:

- not ensuring division was floating point division. Recall if x1, x2, y1, and y2 are all ints the following is still int division:

  ```
  double slope1 = (y2 - y1) / (x2 - x1);
  ```

  You have to do something to make one of the operands a double to cause floating point division to occur:

  ```
  double slope1 = (1.0 * y2 - y1) / (x2 - x1);
  // OR
  double slope1 = (double)(y2 - y1) / (double)(x2 - x1);
  ```

- not guarding against a divide by zero error (which results in a runtime error and the program stopping).

Interestingly if you divide by zero in floating point division it is not a runtime error, rather it yields a value labeled infinity.

Also there is a very clever way of writing the method to remove all these problems. Just cross multiply the slopes. Now there isn't any division, just multiplication and using ints is fine:

```
public static boolean areParallel(int x1, int y1, int x2,
     int y2, int x3, int y3, int x4, int y4) {


     return (y2 - y1) * (x4 - x3) == (y4 - y3) * (x2 - x1);

}
```

4. Suggested Solution:

```
     public static double diffMeanMedian (double[] data)
     {   int middleIndex = data.length / 2;
          double median = data[middleIndex];
          if(data.length % 2 == 0){
               int secondMiddleIndex = middleIndex - 1;
               median = (data[middleIndex] + data[secondMiddleIndex]) / 2;
          }
          double mean = 0;
          for(int i = 0; i < data.length; i++)
             mean += data[i];
          mean = mean / data.length;
          return mean - median;
     }
```

Criteria:

Calculate median: 5 points
Median special cases handled correctly: 5 points

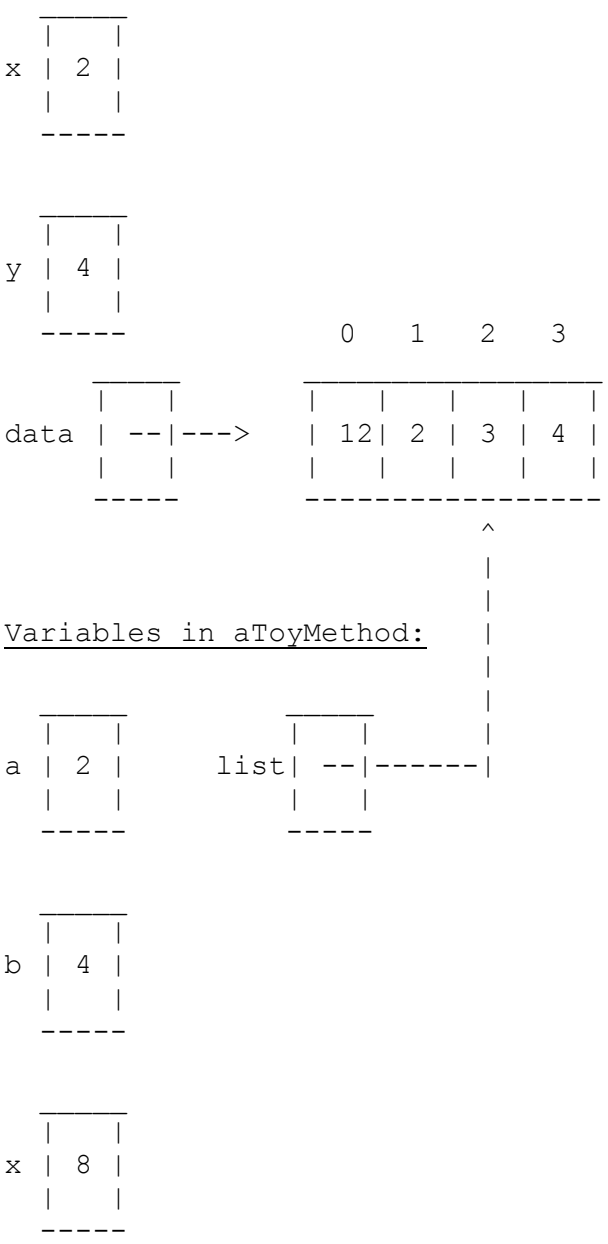
loop through values in array: 6 points
create and keep a running total of values in array: 6 points
calculate mean correctly: 4 points
calculate difference between mean and median: 2 points
return result: 2 points

Commentary:

Students did well on this problem. Common mistakes were:

- not handling the special case with median (lots of off by one and off by two errors on calculating the index of the two values to use to calculate the median.)
- Returning the absolute value of the mean minus the median. The question clearly stated to return the mean minus the median.
- Not accessing elements from the array correctly.

5. Just like we did at the end of the review session in class.



6. Abbreviations A for Always, S for Sometimes, and N for Never

	st.charAt(i) == tgt	count == nth	result == -1
Point A	Sometimes	Sometimes	Always
Point B	Always	Sometimes	Always
Point C	Sometimes	Never	Always
Point D	Always	Sometimes	Always
Point E	Sometimes	Always	Never
Point F	Sometimes	Sometimes	Sometimes