Tip: Use the topics list above to pick topics based on your comfort level. Clearly mark any continuations of your answers on Pages 9 or 10 but do not write on the back. Write your name on each and every sheet.

During the test, you may only consult your 1 sheet of notes and no other resources. You may not use any electronic equipment or other lifeforms. Your answers may refer to any program text supplied in lectures or assignments.

In writing code, you do not need to worry about specifying the proper import statements. Assume that all libraries and packages we’ve discussed are imported in any code you write. There are no penalties for syntax mistakes (typos, etc.) that Eclipse would catch as you type. You may write any helper methods you would like in solving the problems. If you don’t have enough time, assume the helper method and write the rest for partial credit.

First try to answer those questions/parts about which you feel most confident. The questions are partially ordered in order of increasing difficulty. Do NOT spend too much time on one question. Pace yourself @ 75 points/75 minutes.

1. (1 point) Community Standard acknowledgment (signature): ___________________________

2. (7 points) Short answers
   a. (1 pt) What is the type name value (TNV) in the following statement:
      ```java
      char letter = 'x';
      ```
      Type: _____ Name: _____ Value: _____
   b. (1 pt) When you implement a Comparator Interface, you promise to provide _______ method.
   c. (1 pt) You would use a HashMap over TreeMap when you want _______________________
   d. (1 pt) You would use a TreeSet over HashSet when you want _______________________
   e. (1 pt) A hashCode methods that returns a constant leads to many ____________________
   f. (2 pts) When you declare HashMap<K,V>, the hash value is computing using the call back method called _____________ in the class named ________________.

3. (8 points) Big Oh time
   Give the running time of the following code segments as big-Oh in terms of n (tight bound).
   a. (1 pts) Code segment 0
      ```java
      for (int i = 0; i < n; i=i+2) {
         m++;
      }
      ```
      Running time = ______________
   b. (2 pts) Code segment 1
      ```java
      for (int i = n; i > 1; i=i/2) {
         if (i%2 == 0) m++;
      }
      ```
      Running time = ______________
   c. (2 pts) Code segment 2
      ```java
      for (int i = 1; i < n; i++) {
         if (i/2 > 1) m++;
      }
      ```
      Running time = ______________
      ```java
      for (int i = 1; i < n; i=i*2) {
         if (i*2 < N) m++;
      }
      ```
      Running time = ______________
d. (3 pts) Code segment 3
   
   ```
   for (int i = 1; i < n; i++) {
       for (int j = 0; j < n; j=j*2) {
           if (i + j < n) m++;
       }
   }
   ```

4. (8 points) Be There and Be Square

   The picture below shows the first five rows printed by a simple program starting from 0\textsuperscript{th} row.

   ![Diagram of rows](image)

   The pattern continues for n rows so that each row has two more numbers than the previous row.

   Answer the following questions. You may give an expression instead of an exact number for the first two parts (e.g., for example 2, 7, log(7), etc.) Show your work for this question below the answers to get full credit.

   a. (2 points) What is the exact value of the left-most number of the 7\textsuperscript{th} row? 
      
   b. (2 points) What is the exact value of the right-most number of the 10\textsuperscript{th} row? 
      
   c. (2 points) What is the big-Oh of the number of values in the n\textsuperscript{th} row? 
      
   d. (2 points) What is the big-Oh of the rows if the whole structure has n squares? 
      
   Running time =

   \[\text{Expression}\]
5. (6 points) Extend and Implement

Consider the following Interface `IPolygon` and Classes `Shape` and `Square` (in three different files).

Complete the file `Square` with:

a. (3 pts) A constructor that initializes all instance variables that are accessible to `Square`

b. (3 pts) Any methods that `Square` must implement based on the interface `IPolygon`.

```java
// Interface for a regular polygon (all sides are equal)
public interface IPolygon {
    public double getArea(); // Returns the area of a regular polygon
    public double getPerimeter(); // Returns the perimeter of the regular polygon
}
```

```java
// Generic shape... implements perimeter because it is the same for all shapes
public class Shape {
    protected double size, sides;
    protected Color outline;
    public getPerimeter() {
        return size*sides;
    }
}
```

```java
// Square is a regular polygon with 4 sides
public class Square extends Shape implements IPolygon {
    // Code for Square implementation
}
```
6. **(10 points) Common Words**

You are given two arrays for Strings and need to return an int to the number of unique Strings that are common in two arrays. For example:

- If the two input String arrays are "I", "am", "your", "father"} and {"I", "am", "your", "mother"}, then the method returns 3. ("I", "am", and "your" are the 3 unique common words)
- If the two input String arrays are {"I", "am", "what", "I", "am"} and {"I", "am", "I", "am"}, then the method returns 2. ("I" and "am" are the 2 unique common words)

```java
public class CommonWords {
    // Return the number of words common between Strings a and b
    public int count(String[] a, String[] b) {
        // Implementation
    }
}
```
7. **(10 points) TheBestName**

Write a method "`public void sorter(String[] names)`" that takes as input an array of names (Strings) and reorders that array so that it is in sorted order based on the following rules:

A. If two names start with different letters then the one that comes lexicographically (alphabetically) first comes first in the sorted version of `names`. (Amy comes before Yaqub).

B. If two names start with the same letter then the shorter name comes first but if they are the same length then the one that comes lexicographically (alphabetically) first comes first in the sorted version of `names`. (Yaqub comes before Yacoub and Bobbi comes before Bobby.)

- If the input String arrays is
  
  ```
  ```

  then the method reorders it into
  
  ```
  ```

```java
public class TheBestName {

    // Use .sort to reorder the names based on the criteria above
    public void sorter(String[] names) {
```
8. **(25 points) Grades**: Complete the class Grades by implementing the following in the space provided...

a. (5 pts) Complete the constructor by filling in the five blanks with the correct “new” statements and the $i$th element of scores is assigned the value of data[$i$].

b. (5 pts) Write a method public boolean equals(Object other) so that two instances are equal if each element at each position in scores is equal to the corresponding element at that position. (Note: that the other two instance variables are completely dependent on scores). (For example, $\{1, 2, 3\}$ and $\{1, 2, 3\}$ are equal... $\{1, 2, 3\}$ and $\{3, 2, 1\}$ are not equal... $\{1, 2, 3\}$ and $\{1, 2, 3, 4\}$ are not equal...).

c. (5 pts) Write a method public int hashCode() that computes a hash value that is dependent on all the values stored in scores. Any implementation of hashCode is acceptable as long as you use all the values of scores.

d. (5 points) Write a method public void setUnique() that sets the values for unique HashSet so that the each score occurs only once in unique. For example:

   - If scores is $\{80, 90, 100\}$ then unique is $\langle 80, 90, 100 \rangle$
   - If scores are $\{80, 90, 100, 90, 100\}$ then unique is $\langle 80, 90, 100 \rangle$

e. (5 points) Write a method public void setCounts() that sets the values for counts HashMap so that the key is the score and the value is the frequency of that score. The median is the number separating the higher half of scores from the lower half. For example:

   - If scores is $\{80, 90, 100\}$ then counts is $\langle 80, 1 \rangle \langle 90, 1 \rangle \langle 100, 1 \rangle$
   - If scores is $\{80, 90, 100, 90, 100, 100\}$ then counts is $\langle 80, 1 \rangle \langle 90, 2 \rangle \langle 100, 3 \rangle$

```java
public class Grades {
  private ArrayList<Integer> scores;
  private HashSet<Integer> unique;
  private HashMap<Integer, Integer> counts;

  public Grades(int[] data) {
    scores = new ArrayList<Integer>(data);
    unique = new HashSet<Integer>();
    counts = new HashMap<Integer, Integer>();
    for (int i = 0; i < ___scores.length___; i++) {
      ___setUnique();___
      ___setCounts();___
    }
    // Returns true if this .equals other
    public boolean equals(Object other) {
      ___Don't worry about these two methods until you reach Parts d and e___
    }
  }
}
```
// Returns a hashCode that considers ALL values in scores
// Note: that the other two instance variables are completely dependent on scores
public int hashCode() {

}

// Stores the contents of scores into HashSet unique without any duplicates
public void setUnique() {

}

// Stores the counts each score into HashMap unique without any duplicates
public void setCounts() {

}