Statement of integrity: I did not, and will not, break the rules of academic integrity on this exam:

(Signature)

Circle Your Section:

<table>
<thead>
<tr>
<th>Time</th>
<th>PH 219</th>
<th>HO 401</th>
<th>HO 306</th>
<th>PH 403</th>
<th>PH 307</th>
<th>HO 401</th>
<th>HO 306</th>
<th>HO 320</th>
<th>HO 306</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>13 Yan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:25</td>
<td>1</td>
<td>2</td>
<td>11</td>
<td></td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>2:30</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>Artemov</td>
</tr>
<tr>
<td>3:35</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>Artemov</td>
</tr>
</tbody>
</table>

Instructions:

• Read all instructions carefully, and read each problem completely before starting it!
• This test is closed book – no calculators, reference sheets, or any other material allowed.
• Initial or sign each page.
• Conciseness, clarity, and style all count. Show all work and comment code fragments to receive partial credit.
• Arrays are not allowed.
• You may not alter the structures surrounding blanks and boxes. Only one statement or expression per blank!
• Use the backs of pages if you need more space or scrap. You may request additional sheets from a proctor.
• If you supply multiple answers, we will grade only one.

Core Points:

1. _______ (27 points) _________
2. _______ (15 points) _________
3. _______ (37 points) _________
4. _______ (21 points) _________

Total: _______ / (100 points) _________

Bonus Points: _______ / (9 bonus points) _______
Problem 1 [27 points] Nested Loops

Write a program that reads a non-ascending sequence of even integers. Assume that the user enters only even integers. For each integer \( n \), print a row of \( n \) stars (*). Your program must center each row of *s underneath the top row. Do not print any blank spaces at the beginning of the first row. A non-positive integer terminates the sequence. You must have comments for major variables, each loop, and a comment describing exactly how many blank spaces printed on each row before the *s.

Example: For the input sequence 10 4 4 2 -2, the output would be

```
**********
****
****
**
```

```java
public class Problem1 {
    public static void main(String[] args) {
        TokenReader in = new TokenReader(System.in);
```
Problem 2  [15 points] Tracing

Fill in the box scope diagram below for the following code. For non-reference variables (int, double, boolean, etc.), write the value in the corresponding box. For reference variables, write null or draw an arrow from the reference variable to the object box. Fill in all default values for instance variables. If a variable has undetermined contents, draw a question mark (“?”). To update the contents of a variable, cross off the old contents and write the new contents nearby. “Cross off” means “draw a single, tidy X over the value or on the arrow.” You MUST use only the boxes we’ve drawn for you.

Note: We have provided a surplus of object boxes. You must cross off the unused object box(es). See the next page for spare boxes. You must indicate which diagram we should grade! Otherwise, we will choose just one to grade.
Spare Boxes for Problem 2

String Class

main(args)

G1

W1

W2

args

Main Class

Gorph Class

Sputty

s

g

w

Gorph Class

s

g

w

Whappy Class

x

g

s

g

x

g

Whappy Class
Problem 3  [37 points] OOP: Design a class; See the figure in Problem 4 for a picture

Fill in the 3 blanks and 2 boxes below to complete the code for a class Worker to represent people working in a company. Each Worker has exactly the following instance variables – do not define additional instance variables:

- **id**: her or his ID number (an integer).
- **boss**: a reference to the Worker, if any, who is her or his boss.

Each Worker has exactly the following methods – do not define additional methods or constructors:

- **owner()**: return the reference to the owner of the company, the “top-most” Worker who has no boss.
- **describe()**: return a String that contains the ID, ID of the boss, and ID of owner. For the case of the owner, use "NO BOSS" in place of the boss’s ID. You must include labels for each item! For example, a returned String might be "ID: 10, ID for boss: NO BOSS, ID for owner: 10".

Notes: This approach means a Worker does not "know" its underlings – there are no links “down.”

```java
class Worker {
    _____________________________ ; // ID
    _____________________________ ; // reference to boss

    // Return reference to owner. For full credit, use a loop!
    Worker owner() {
    }

    // Return a String containing IDs of self, boss, and owner
    _____________________________ describe() { // Fill in the return type!
    }
}
```
Problem 4 [21 points] OOP: Using a class

This problem uses a Main Class called Problem4 that accesses class Worker from Problem 3. Assume that class Worker has been completely and correctly defined. The main method instantiates Workers and establishes the links that relate Workers and their bosses. Let the notation \#w represent the Worker whose ID is w. The figure below shows the links for companies A, B, and C using Worker IDs. Thus, the owner of company A is \#60, the Worker whose ID is 60. Also, \#60 is the boss of \#6, who is the boss of \#2 and \#3:

![Diagram of company A and B hierarchy](image)

Your code merges companies A and B into company C. Note the newly hired Worker \#30 in company C.

Fill in the 7 blanks to complete method main, below. Assume different companies have different Workers, and hence, different owners. Thus, two companies are the same if they have the same owner. Read the comments!

```java
public class Problem4 {
    public static void main(String[] args) {
        Worker sixty; // Worker \#60
        Worker five;  // Worker \#5
        Worker x; Worker y; // References to unspecified and different Workers

        // Establish companies A, B, and others.
        // Make $x$ and $y$ refer to instances of different Workers.
        Code That We’re Not Showing, So Don’t Ask!

        // Use $thirty$ to establish company C as shown in the figure, above.
        // Use variables $sixty$ and $five$, which refer to \#60 and \#5, resp.
        Worker thirty = ________________________; // Instantiate \#30
        ______________________________________; // Set ID of \#30 to 30
        ________________________________________; // Link \#30 and \#60
        ________________________________________; // Link \#30 and \#20
        // Print a description of the owner of the company where $x$ works
        ________________________________________;

        // If $y$ and $x$ work for the same company, do nothing. Otherwise,
        // print a description of the owner of the company where $y$ works.
        ________________________________________;
    }
}
```

Use instance variables id and boss and methods owner() and describe() from class Worker in Problem 3.
**Bonus:** [9 bonus points] *Don’t work on this until 100% done with CORE problems!*  
Do NOT attempt any of these bonus problems until you have completed **all** the core problems. Do NOT answer any bonus problems that you’re not sure how to answer:  
+ (Mostly) correct answers will be rewarded with bonus points.  
0 Unanswered bonus problems will be neither penalized or rewarded.  
– (Mostly) incorrect answers will be penalized bonus points.

**B1)** [2 bonus pts] How many total possible Project points are there? ________  
How many of those are required for a perfect Project score? ________

**B2)** [2 bonus pts] Prof. Yan posted an IMPORTANT message on the newsgroup about a "secret weapon" for loops.  
What was his or JavaLive’s **two-word** term for the secret weapon? ____________ ________________  
Let the two blanks ________ ________ represent that two-word term. Now, using that term, what is a suitable ________ ________ for a loop to fill an empty tank with 100 liters of water in 50 seconds? Assume that variable time measures the number of elapsed seconds.

**B3)** [2 bonus pts] Write method **owner()** recursively, i.e. without using loops. (Recall: you were required to use a loop for your solution in Problem 3.)

**B4)** [2 bonus pts] Recall that class **Parthon** on Project 4 has instance variables **parent, youngestChild, olderSibling** and method **describe**. (You don’t need the other instance variables and methods for this problem.) Write a recursive method **dtd()** to describe all the teenage descendants of a **Parthon**. You may also describe some or all of the **Parthon**’s siblings’ descendants. The descendants of a **Parthon** are itself, its children, its children’s children, etc. Teenagers have ages between 13 and 19, inclusive.

**B5)** [1 bonus pt] Suppose we add to class **Worker** a constructor **Worker(i, b)** that initializes its **id** to **i** and its **boss** to **b**. Draw the picture of worker relationships established by the code below. (Draw in the style of the figure in Problem 4, not in the style of box/scope diagrams.)

```java
new Worker(1,
    new Worker(2,
        new Worker(3,
            new Worker(4, null)
            ).boss
        )
    ).boss
);
```