INSTRUCTIONS

This is a closed book and note examination, except for one 8 1/2 x 11 inch paper sheet of notes, both sides. There is no interpersonal or other communication or discussion (except with the prof. or proctors) or sharing of information. Cell phones or any electronic devices (other than a calculator) are strictly forbidden! You must remain in the exam room at all times. If you need to leave the room, come to the front, leaving your paper and ask permission. You may take NOTHING with you to leave the room (and that includes cell phones!). Failure to abide by any of these terms will result in a grade of zero for the exam.

You must show a picture ID to the instructor or TA when you turn in your exam.

NAME ________________________________________________________________

NET ID ________________________________________________________________

LAB SECTION (circle one):

Mon 12:35  Mon 4:40  Tue 2:45  Wed 1:40  Wed 4:15
Mon 1:40   Tue 1:15  Wed 11:30  Wed 2:45  Thurs 2:45

------- For scoring use only. Do not write below this line -------------
Section 1 – Multiple choice and short answer (13 questions, 3 points each)
Select the best answer to the question from the choices provided.

1. A Java comment that
   ○ begins with /** (instead of */ or // ),
   ○ can have tags such as @param, @return, and @author, and
   ○ can be transformed automatically to a Web page of documentation
   *is specifically called the following kind of comment:*
     a. javadoc
     b. DrJava
     c. Version History
     d. Optional Documentation

2. An array of int variables referred to by A starts out so when the code
   \[
   \text{for( int } i = 0; i < A.length; i++ ) \text{ System.out.print(" " + A[i]);}
   \]
   is run, the computer prints
   \[
   10 \ 20 \ 30 \ 40
   \]
   After printing this, suppose the computer performed the code:
   \[
   \text{A[ 1 ]} = \text{A[ 0 ]}; \text{ Hint: Name of ticket = value to write on it;}
   \text{A[ 2 ]} = \text{A[ 1 ]};
   \text{A[ 3 ]} = \text{A[ 2 ]};
   \]
   After these 3 lines of code, suppose the computer performed again
   \[
   \text{for( int } i = 0; i < A.length; i++ ) \text{ System.out.print(" " + A[i]);}
   \]
   What is printed this time? SHOW the 4 element array with YOUR WORK
   to figure out the answer written in the array you show BELOW!!!
   \[
   \begin{align*}
   &a. \ 10 \ 10 \ 20 \ 30 \\
   &b. \ 40 \ 30 \ 20 \ 10 \\
   &c. \ 20 \ 30 \ 40 \ 40 \\
   &d. \ 10 \ 10 \ 10 \ 10 \\
   &e. \ 40 \ 40 \ 40 \ 40
   \end{align*}
   \]

3. According to the Mad Ph.D. video, explain briefly in English what the part
   "new Happy_House()" makes the computer do, of the Java code line:
   \[
   \text{Happy_House \ mad_drs_house = new Happy_House( );}
   \]

4. According to the Mad Ph.D. video, explain briefly in English what the part
   "mad_drs_house =" makes the computer do:
5. The **new** operator  
   a. Makes a new class.  
   b. Calls a constructor first, and second makes a new object, as described in answer c.  
   c. Makes an object with fields (variables) given in the class definition; it uses the class definition like a blueprint. A constructor is called second, **after** the object is made or built.

6. The **new** operator  
   a. Must be used to make an array. (Code like \{ 1, 2, 3 \} is shorthand for the **new** operation plus 3 assignments.)  
   b. Is never used to make an array. An array is made with code like  
      ```java
      int A[ ];
      ```
   c. Is sometimes used to make an array, but Java arrays can be made without it other times.

7. Recall the vignette project, Project 2. The three RGB intensities used to recolor a **Pixel** were weighted averages computed as Java **double** values (number values with decimal points.) However, you had to cast them to **int** values when you used them in the **Color** constructor, in code like  
   ```java
   new Color( (int) ( weighted average expression ), ... );
   ```

   **Why were the casts, to change double to int, necessary?**  
   a. The **Color** constructor requires 3 **int** parameters, not 3 **doubles**.  
   b. **Ints** are exact, so we use them to make more precise colors.  
   c. Actually, the casts are unnecessary; any number in the 0.0–255.99 range is OK.  
   d. A color value with a fractional part like 33.35 will look gritty.

8. Compare the Java **boolean** type with the Java **int** type.  
   a. The **boolean** type has only two values but the **int** type has billions.  
   b. Both the **boolean** type and the **int** type have billions of values.  
   c. The **int** type has only two values but the **boolean** type has billions.  
   d. Both the **boolean** type and the **int** type have only two values.  
   e. A type like **boolean** or **int** has nothing to do with values, it's only what you must write to avoid syntax errors and has no other significance.
9. Weighted averages were used in making the vignette. In each weighted average, (1) two numbers were averaged together, and (2) some weight \( w \), \( 0.0 \leq w \leq 1.0 \) was used. \( w \) had to be double, since it's usually between 0 and 1. Refer to the figure below, which shows one of the many, many Pixels to get a new Color, and two relevant distances HD and R.

Explain in English: What were the two numbers averaged together?

10. How many of those weighed averages were computed for each (single) Pixel?
   a. 1
   b. 2
   c. 3
   d. more than 3

11. Again referring to the vignette project and the diagram below, what was the weight \( w \)? Give a formula and/or explain how \( w \) is computed from information shown in the figure.

But, you are responsible for the relationship between averages and blurring!
The `chromakeyBlue` method from GE that you coded into the `Picture` class, and perhaps modified to better handle our green background, begins:

```java
/**
 * Method to do chomakey using a blue background
 * @param newBg the new background image to use to replace
 * the blue from the current picture
 */
public void chromakeyBlue(Picture newBg)
```

The `Picture` object it was called on was of you in front of a green screen. `newBg` refers to a separate, scenic background. The method uses one `if` statement.

12. Explain, as precisely as you can in English, the purpose of that `if` statement.

13. In the use case of Project 4 where the user specified the width and height of the final result `Picture`, it was OK to (A) make the final result `Picture` first, and second, (B) copy each `Picture` into the already-made final result `Picture` when each was given to `addPicture( ... )`. So, the computing steps (A) and (B) could be done in the order AB. However, in the other use case, the final result `Picture`’s width and height were not known until after all the added `Pictures` were stored in the computer. We list four of the computations for that other use case, listed in an order ABCD that is impossible to carry out:

A: Make the final result `Picture`.
B: Copy the added `Pictures` into the final result `Picture`.
C: Calculate the width and the height that the final result `Picture` should have.
D: Make the array that will be used to save and store the references to the added `Pictures`.

Again, ABCD is an impossible order. Write below the order (a permutation of A, B, C, D) in which the above four computations can be done:

```
---------------------
```

**Corresponding projects in 2012 were different.**
This flowchart for for is faulty in several ways.

(6 points)
Redraw it correctly below:

It is for
for ( INIT ; TEST ; INCRM )
{  
    BODY
}

(5 points) Below, draw the flowchart for
if ( TEST )
{ IF-BODY }
else
{ ELSE-BODY }

In 2012, the emphasis was on while statements, and exactly what they make the computer do.

Also, study if ( ... ) { ... } and
if ( ... ) { ... } else { ... }
very carefully!
Your job is to add a method to the Picture class that will put a 45 degree upper-left to lower-right black single pixel width stripe onto the Picture referred to by this. For the full 30 points credit, it must work on tall, skinny Pictures as well as short, wide Pictures. You will get 20 points if it works correctly on square Pictures but crashes on certain non-square ones. Here is what the result should look like on three shapes of Pictures:

The stripe must begin at the upper left hand corner. Make it by blackening all the Pixels whose locations have the form (0,0), (1,1), (2,2), etc. For full credit, you will have to make sure it doesn't access any locations outside the Picture. Minor syntax errors will be forgiven as long as your code clearly demonstrates accurate understanding of the problem, solution, and Java programming.
The purpose of this question is for you to demonstrate how to restructure a program to make a shorter and more clear program by (1) removing repeated code and putting just one copy of removed code in a added parametrized method and (2) calling that method from different places, with different parameters.

Somebody put the copy method into the Picture class. It copies the given Picture srcP into "this" Picture at the given X location int xWhere. But, the boolean parameter atTop determines whether the copy is put at the top or at the bottom. See the examples below. (srcP is the filled in rectangle.)

```java
public void copy(Picture srcP, int xWhere, boolean atTop)
{
    if( atTop == true )
    {
        for(int y = 0; y < srcP.getHeight(); y++)
        {
            for(int x = 0; x < srcP.getWidth(); x++)
            {
                Pixel p = this.getPixel(xWhere + x, y);

                p.setColor(srcP.getPixel(x, y).getColor());
            }
        }
    }
    else
    {
        for(int y = 0; y < srcP.getHeight(); y++)
        {
            for(int x = 0; x < srcP.getWidth(); x++)
            {
                Pixel p = this.getPixel(xWhere + x, y+(this.getHeight()-srcP.getHeight()));

                p.setColor(srcP.getPixel(x, y).getColor());
            }
        }
    }
}
```

The answer goes on the next page....

Page 8 of 10
Demonstrate that you can make the program better by (A) making a new method containing ONLY ONE doubly-nested LOOP, to be called from two different places in copy, with suitable parameters, and (B) replacing the two doubly-nested for loops with two calls to the copy method.

Your answer to (A) goes in the method below:

```java
private void generalCopy(Picture srcP, int xWhere, int yWhere)
{
    //YOU code the doubly-nested loop to do the copying!!
    //Hint: Copy a few lines from page 8 with small changes.
}
```

Your answer to (B) goes in the method below:

```java
public void copy(Picture srcP, int xWhere, boolean atTop)
{
    //YOU rewrite the copy method from above so it has no loops but it calls generalCopy from two places with the correct parameter values.
    if( atTop == true )
    {
        
    }
    else
    {
        
    }
}
```
(10 points) Harder array programming problem.

Create a static (that is, class) method named `rotateRight`. Its one parameter must be an `int` array (reference): `rotateRight(int A[ ])` It should not return a value (so declare its return type `void`), but it should re-arrange the values stored in the array elements.

The rightmost (last) element should be moved to the beginning of the array, and the other elements should be shifted one position to the right.

For example, if `AA` refers to the length 4 array containing the numbers below,

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>35</td>
<td>9</td>
<td>17</td>
</tr>
</tbody>
</table>

and the method call `rotateRight( AA )` is done, the `rotateRight` method should return with array `AA` rearranged and containing:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>13</td>
<td>35</td>
<td>9</td>
</tr>
</tbody>
</table>

Hint 1: It must work an ANY length array (not just 4), so it must (1) get the array length with `A.length` and (2) use a loop. Hint 2: Take care of the last element first and make sure your code does not make the computer lose the values that were originally from the first and last elements; DON'T use `A[0] = A[A.length-1]`; as your first operation because that makes the computer lose the old value from `A[0]`!