University of Illinois at Urbana-Champaign
Department of Computer Science

Second Examination

CS 125 Introduction to Computer Science
90 minutes permitted

First name: _____ Last name: _____

NetID: _ _ _ _ _ _ _ _ @ illinois.edu
(please write legibly)

Discussion Section: AY_____
(We will return your exam manuscript to you in this section)

```
AYA Tues  9:00am-10:50
AYB Tues 11:00am-12:50
AYC Tues  1:00pm- 2:50
AYD Tues  3:00pm- 4:50
AYE Wed   9:00am-10:50
AYF Wed  11:00am-12:50
AYG Wed   3:00pm-04:50
```

• This is a closed book and closed notes exam. No electronic aids are allowed. You must turn this exam booklet in before leaving

• You should have 9 sheets total including one scratch sheet The last sheet is scratch paper; you may detach it while taking the exam, but must turn it in with the exam when you leave.

• Carefully read the specification given for each class. You may not write, or assume the existence of, any unspecified additional methods or constructors. Also, you are not allowed to use the break, continue, or switch statements on this exam.

• Unless we say otherwise in the specific problem, you can assume all values entered by the user will be acceptable input for that program.

• When you write code, you may use a shorthand for System.out and TextIO input and output methods provided it is obvious to the graders which method you are using. For example it is acceptable to use Sopln in place of System.out.println and to use Sopt in place of System.out.print Likewise, you can use T.rlnI(), T.rlnC(), and T.rlnD() in place of TextIO.readlnInt(), TextIO.readlnChar(), and TextIO.readlnDouble().

• For full marks correct syntax is required: Ensure all statements include a semicolon and the correct use of upper/lower case, single quotes and double quotes. However a syntactically correct but flawed program will earn a low score i.e. semantics and correctness is more important than minor typos and syntax errors.

Problem	Points	Score	Grader
1	15		
2	16		
3	15		
4	20		
5	16		
6	18		
Total	100		

1. Short Answers – 15 points (3 points each)

(a) What is printed by the last line of the following Java code? Assume that the constructor initializes the private instance variables and that no compile or runtime errors occur. You do not need to write an explanation.

```
Robot r1, r2, r3;
r1 = new Robot("Huey",  2,3);
r2 = new Robot("Louie", 5,4);
r3 = new Robot("Louie", 5,4);
r1 = r2;
r2 = r3;
r3 = r1;
System.out.print(r1 == r2);
```

Write exactly the program output here and nothing else: _____

(b) Class B extends class A. Both classes define a *hide*() method. Which *hide* method is used in the following code snippet? [3 points]

```
A x = new B();
x.hide();
```

Write A or B here: _____

(c) The behavior of the above code in (b) is best described as an example of ...

A) Primitive types

B) Operator overloading

C) Inference

D) Polymorphism

E) Type checking

Your answer: _____

(d) True or False: Objects of Java's String class are immutable?

Your answer: _____

(e) Briefly explain how the following statement can be true, "I've just finished writing a new class. All instance variables of my class are private but the objects of this class are not immutable." You may create an example class as a demonstration or provide a short written explanation.

2. Be careful when comparing strings – 16 points

Complete the *FriendsList* class by implementing the *add* method and the three other methods described below.

```
public class FriendsList {
  private String[] list  = new String[0];

  public void add(String name) {
//Allocate a larger array:


//Copy the old list contents :




//update this.list to reference the new array:



 //Don't forget to add the new friend:


}
/* Write three more public instance methods for FriendsList:
 - getSize to return the number of friends as an integer
 - getFriend that takes an integer parameter 'i' and returns a
string, the friend at index i in the array i.e. 0<=i<list.length
 - is007 that takes no parameters and returns a boolean. Return
true if any of the friends' names in this list are "James Bond"
(exact case, quotes not included), false otherwise. Assume all
entries are non-null. */
//getSize:



// getFriend:




// is007:









} // end class FriendsList
```

3. I took CS125 and all I got was this startup company and nerdy friends – 15 Points

You are working on a photo manipulation program for a camera phone. Your startup partner has written the following code that processes two source images (*srcA, srcB*) and returns a new output image. Images are represented as a 2D array. The first array index represents the column, the second the row. Thus *array*[x][y] represents the red-green-blue pixel color at (x,y) encoded as a single integer.

```
1   public static int[][] mystery(int[][] srcA, int[][] srcB) {
2      int width= srcA.length, height = srcA[0].length;
3      int[][] output = new int[width][height];
4      for (int x = 0; x < width; x++) {
5         if(x%2 ==0) output[x] = srcA[x];
6         else output[x] = srcB[x];
7      }
8      return output;
9   }
```

(a) You test the mystery function using several pairs of images. For the two 3x4 source images below, shade the resulting output image.

srcA	x=0	x=1	x=2
y=0			
y=1	■	■	■
y=2			■
y=3			■

srcB	x=0	x=1	x=2
y=0	■		■
y=1		■	
y=2	■		■
y=3		■	

output	x=0	x=1	x=2
y=0			
y=1			
y=2			
y=3			

(b) When tested with two particular test images for *srcA srcB*, the function fails with an *IndexOutOfBoundsException* at line 6. Briefly explain what must be unusual about this particular image test pair to cause this problem.

(c) Later, you test on a real phone. The function fails with an exception at line 2. Suggest a possible parameter value that would cause the function to fail at line 2.

(d) You change the mystery function by replacing lines 5 and 6 with new code. Write your new code below such that the output is identical to the *srcA* image except that any *srcA* pixels that are black (value 0) are replaced by *srcB*'s color at the same x,y position. Your implementation can ignore the special-case problems described in (b) and (c).

4. World of Goo or Battleships? – 20 Points

Complete the class Boat according to the following specification. Do not create any other additional constructors, instance or class methods or instance or class variables – only create the constructors, methods and instance variables specified below.

1. Three private integer instance variables, *x*, *y* and *count*.
2. A public constructor that takes two integers to set the boat's *x,y* position and also sets the initial value of *count* to 3.
3. A public copy constructor that takes a reference to an existing boat. Copy the given boat location and count.
4. A public method named 'turn' that takes a reference to a *Point* object (see code on the right) and returns a boolean. If the point's position equals the boat's position, decrement *count* and return true. Otherwise return false.
5. A public method named 'isAlive' that takes no parameters and returns a boolean. Return true if *count* is zero or positive, false otherwise.

```
public class Point {
  public int x;
  public int y;
}
```

The complete *Point* class

```
public class Boat {
// Write your implementation here:
```

```
} // end class Boat
```

5. We all live in a yellow submarine – 16 Points

Create a new class named 'Submarine' that extends the Boat class (defined in Q4) according to the following specification. Do not create any other additional instance or class methods or instance or class variables – only create the methods and instance variables specified below.

1. Submarine has one additional private boolean instance variable '*submerged*'.
2. A public constructor that takes no parameters, initializes the superclass's position to (5,6) and initializes *submerged* to true or false with equal probability. Hint: Math.random() is useful here.
3. Override the boat's method 'turn': If the submarine is submerged, the turn method immediately returns false, otherwise the submarine behaves like a boat i.e. return the result of calling the boat's existing 'turn' method.
4. A public method named 'toggle' that takes no parameters and does not return anything. Instead it toggles the *submerged* variable (change false to true and true to false).

6. Should class methods be static? – 18 Points

1. Create a public class method named 'create' that takes an integer '*n*' as a parameter – the number of boats to create – and returns an array of boat objects (the Boat class is specified in Q4). Create the boats at positions (0,0), (1,1), (2,2)... etc. Do not create any submarines.
2. Create a public class method 'boatcopy' that takes an array of boat objects and returns a new array of boats. Perform a deep copy.
3. Use BoatUtil methods: Complete the main method below which is in a DIFFERENT class, BattleshipGame. Create an array of 7 boats using BoatUtil's *create* method; Create a deep copy of this array using *boatcopy*; Then call *turn* on the first boat of the deep copy with the parameter being a Point object (see Q4) that represents the origin (x=0,y=0).

BoatUtil.java

```
class BoatUtil {
// create:









// boatcopy:









} // end of class
```

BattleshipGame.java

```
public class BattleshipGame {
  public static void main(String[] args) {









 }
} // end of class (END OF EXAM)
```

Empty Page

Scratch paper