CIS 110 — Introduction to Computer Programming
Spring 2015 — Final Exam

Name: _____
Recitation # (e.g., 201): _____
Pennkey (e.g., eeaton): _____

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

_____ _____
Signature Date

Instructions:

- **Do not open this exam until told by the proctor**. You will have exactly 110 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts.**
- Food, gum, and drink are strictly forbidden.
- **You may not use your phone or open your bag for <u>any</u> reason**, including to retrieve or put away pens or pencils, until you have left the exam room.
- This exam is *closed-book, closed-notes, and closed-computational devices*.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All answers must be written on the exam booklet.
- All code must be written in proper java format, including all curly braces and semicolons. You do not need to comment your code, but may do so if it helps clarifies your answer.
- Do not separate the pages. If a page becomes loose, re-attach it with the provided staplers.
- Staple all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper, please use the backs of the exam pages or the extra pages provided at the end of the exam. **Clearly indicate on the question page where the graders can find the remainder of your work (e.g., "back of page" or "on extra sheet").**
- Use a pencil, or blue or black pen to complete the exam.
- If you have any questions, raise your hand and a proctor will come to answer them.
- When you turn in your exam, you may be required to show ID. **If you forgot to bring your ID, talk to an exam proctor <u>immediately</u>.**
- We wish you the best of luck.

Scores: [For instructor use only]

Section 0: Cover Page		1 pt
Section 1: Miscellaneous		10 pts
Section 2: Sorting		12 pts
Section 3: Tracery		15 pts
Section 4: Recursion		8 pts
Section 5: Debugging		15 pts
Section 6: Linked Lists		18 pts
Section 7: Data Structures		16 pts
Total:		**95 pts**

SECTION 0: COVER PAGE INFORMATION (1 pt)

- Check that your exam has all 11 pages (excluding scratch paper).
- Write your name, recitation number, and PennKey (username) on the front of the exam.
- Sign the certification that you comply with the Penn Academic Integrity Code.

SECTION 1: MISCELLANEOUS (10 pts total)

1.1. **(2 pts)** Which of the following is **NOT** a primitive type in Java? (Choose all that apply)

A. `int` B. `int[]` C. `double` D. `String` E. `char[]`

1.2. **(2 pts)** Which of the following are examples of invoking static methods? (Choose all that apply)

A. `Math.sqrt(2.0)`
B. `in.readString()`
C. `java NBody 1000000 1000 planets.txt`
D. `"one2three".charAt(3)`
E. `p = new Person()`

1.3. **(2 pts)** Assume you're sorting the `String[] args` array for a java program that requires exactly ten (10) arguments. Which sorting algorithm should you use? (Choose the best answer)

A. insertion sort
B. selection sort
C. merge sort
D. It doesn't matter. Since the array size is small, these algorithms take roughly equal time.
E. None of the above, since the array is already sorted.

1.4. **(2 pts)** Assume you're sorting an array of all passport numbers in the United States. Which sorting algorithm should you use? (Choose the best answer)

A. insertion sort
B. selection sort
C. merge sort
D. It doesn't matter. Since the array size is small, these algorithms take roughly equal time.
E. None of the above, since the array is already sorted.

1.5. **(2 pts)** Which of the following can static methods NOT do? (Circle all that apply)

A. Refer to any instance variables of the class
B. Refer to *this*
C. Refer to other static variables
D. Be included in a class with non-static methods.
E. None of the above

SECTION 2: SORTING (12 pts total; 2 pts each)

Recall that insertion sort and selection sort separate the array into two portions: the left portion is always in ascending sorted order, and the right portion is not. Each step of the algorithm shifts the boundary one array position to the right, sorting as it goes. In contrast, mergesort recursively subdivides the array and then each step merges two subarrays back in sorted order.

Assume that we're sorting the array `{9, 1, 8, 0, 2, 6, 3, 0}` using insertion sort, selection sort, and mergesort. Match each of the arrays from the right column with the appropriate algorithm and step number from the left column. Letters may be used more than once. The first one has been done for you as an example.

Ex. Insertion sort after two (2) steps ___J___

2.1 Insertion sort after four (4) steps _____

2.2 Selection sort after two (2) steps _____

2.3 Selection sort after three (3) steps _____

2.4 Mergesort after two (2) merges _____

2.5 Mergesort after three (3) merges _____

2.6 Mergesort after six (6) merges _____

A. `{0, 0, 1, 2, 3, 6, 8, 9}`

B. `{0, 0, 1, 2, 9, 6, 3, 8}`

C. `{0, 0, 1, 9, 2, 6, 3, 8}`

D. `{0, 0, 8, 9, 2, 6, 3, 1}`

E. `{0, 1, 8, 9, 0, 2, 3, 6}`

F. `{0, 1, 8, 9, 2, 6, 0, 3}`

G. `{0, 1, 8, 9, 2, 6, 3, 0}`

H. `{1, 8, 9, 0, 2, 6, 3, 0}`

I. `{1, 9, 0, 8, 2, 6, 3, 0}`

J. `{1, 9, 8, 0, 2, 6, 3, 0}`

K. `{9, 1, 8, 0, 2, 6, 3, 0}`

L. `None of the above`

SECTION 3: TRACERY (15 points total)

What will happen when `MonstersInc` is run? You should assume that the `rooms[]` arrays are randomly generated as follows (and in this order):

```
{1, 1, 2, 5, 8}
{2, 9, 1, 1, 9}
{6, 6, 5, 2, 7}
```

These are the only random values known; you must solve this problem using only this information.

```
public class Monster {
  public int strength; //strength between 1 and 10
  public String name;
  public int score = 0;
  public Monster(int strength, String name) {
    this.strength = strength;
    this.name = name;
  }
}

public class MonstersInc {
  private static Monster[] employees;
  private static int totalScreams;
  private static int screamNeeded = 100;

  public static void hireEmployees(String[] names) {
    employees = new Monster[names.length];
    for (int i = 0; i < employees.length;) {
      int random = (int)(Math.random() * 10);
      if (random > 6) {
        employees[i] = new Monster(random, names[i]);
        i++;
      }
    }
  }

  public static void work(int[] rooms) {
    for (int i = 0; i < employees.length; i++) {
      totalScreams += employees[i].strength;
      employees[i].score += rooms[i];
    }
  }

  private static void exch(Monster[] a, int i, int j) {
    Monster swap = a[i];
    a[i] = a[j];
    a[j] = swap;
  }

  public static void rank() {
    for (int i = 1; i < employees.length; i++){
      for (int j = i; j > 0; j--){
        if (employees[j - 1].score < employees[j].score) {
          exch(employees, j-1, j);
        } else break;
      }
    }
  }
```

```
  public static void printList() {
    for (int i = 0; i < employees.length; i++) {
      System.out.println(employees[i].name + " has a score of "
                         + employees[i].score);
      }
  }

  public static void main (String[] args) {
    String[] names = {"Boo", "Joe", "Mike", "Sully", "Lily"};
    hireEmployees(names);
    printList();           // LOCATION A

    while(totalScreams < screamNeeded) {
      int[] rooms = new int[names.length];
      for (int i = 0; i < rooms.length; i++) {
        rooms[i] = (int)(Math.random() * 10);
      }
      work(rooms);
    }
    rank();
    printList();           // LOCATION B
  }
}
```

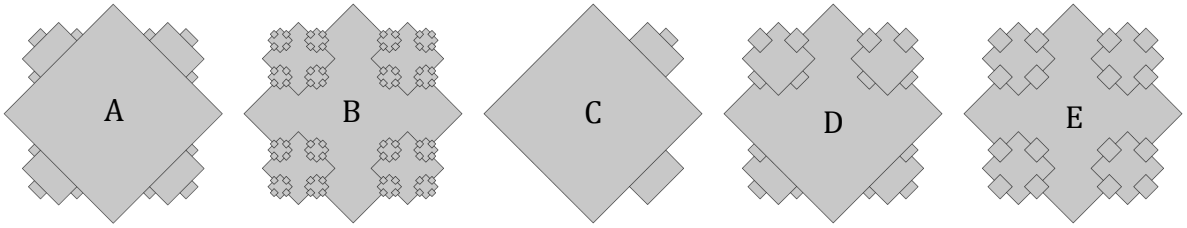
3.1 (5 pts) What is the output of the program at LOCATION A? (print clearly)

3.2 (10 pts) What is the output of the program at LOCATION B? (print clearly)

SECTION 4: DIAMOND THIEF! (8 pts total; 2 pts each)

Background: All of the TAs competed to make the best submission for the Art assignment. However, some TAs simply imitated an example from the textbook. We would like to find out who they are, but unfortunately their code and artwork got mixed up.

Question: Match the functions with their appropriate output (A-E below). All functions were run via `recursionN(0.5, 0.5, 0.5, 3)`. If a function doesn't match any of the outputs, write "NONE".



4.1
```
static void recursion1(double x, double y, double size, int n) {
    if (n < 0) return;
    drawDiamond(x, y, size);
    recursion1(x + size / 2, y + size / 2, size / 3, n -= 1);
    recursion1(x - size / 2, y + size / 2, size / 3, n -= 1);
    recursion1(x + size / 2, y - size / 2, size / 3, n -= 1);
    recursion1(x - size / 2, y - size / 2, size / 3, n -= 1);
}
```

4.2
```
static void recursion2(double x, double y, double size, int n) {
    if (n <= 0) return;
    drawDiamond(x, y, size);
    recursion2(x - size / 2, y + size / 2, size / 3, n - 1);
    recursion2(x + size / 2, y - size / 2, size / 3, n - 1);
    recursion2(x + size / 2, y + size / 2, size / 3, n - 1);
    recursion2(x - size / 2, y - size / 2, size / 3, n - 1);
}
```

4.3
```
static void recursion3(double x, double y, double size, int n) {
    if (n <= 0) return;
    recursion3(x + size / 2, y - size / 2, size / 3, n - 1);
    recursion3(x - size / 2, y + size / 2, size / 3, n - 1);
    recursion3(x - size / 2, y - size / 2, size / 3, n - 1);
    recursion3(x + size / 2, y + size / 2, size / 3, n - 1);
    drawDiamond(x, y, size);
}
```

4.4
```
static void recursion4(double x, double y, double size, int n) {
    if (n <= 0) return;
    recursion4(x + size / 2, y - size / 2, size / 3, n - 1);
    recursion4(x - size / 2, y - size / 2, size / 3, n - 1);
    drawDiamond(x, y, size);
    recursion4(x + size / 2, y + size / 2, size / 3, n - 1);
    recursion4(x - size / 2, y + size / 2, size / 3, n - 1);
}
```

SECTION 5: DEBUGGING (15 points total)

Find and correct the remaining errors in the code on the next page. The first bug has been found for you, and is listed as an example in the table. Some of the bugs may cause incorrect behavior rather than syntax or runtime errors.

The number of remaining errors is **unknown**; you may not need to use all rows in the table below.

In writing your answer, list the line number(s) followed by the corrected code. You may assume that the author of the code has given the fields and methods names that reflect their intention.

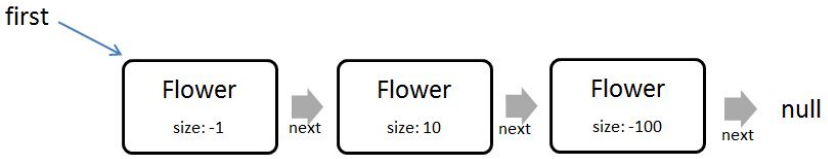
Hint: Each correction should require one line of code or less.

Line Number	**Correction**
03	Change "string" to "String"

```
01    public class GuestList {
02      private Guest guests;
03      private string title;
04      private int numGuests = 0;
05      private int listCapacity;
06
07      public class Guest {
08        public String name;
09        public int age;
10        public Guest next;
11        public Guest(String name, int age, Guest next) {
12          this.name = name;
13          this.age = age;
14          this.next = next;
15        }
16      }
17
18      /*
19       * listCapacity is guaranteed to be a positive integer
20       * listTitle is guaranteed to be non-null
21       */
22      public GuestList(String listTitle, int listCapacity) {
23        title = listTitle;
24        this.listCapacity = listCapacity;
25      }
26
27      public int getNumGuests() {
28        return numGuests;
29      }
30
31      // name is guaranteed to be non-null, and age will be valid
32      public void addGuest(String name, int age) {
33        if (getNumGuests() > listCapacity) {
34           throw new RuntimeException("Capacity reached. Cannot add new guest");
35        }
36        guests = new Guest(name, age, guests);
37      }
38
39      // Determine whether the list contains someone with a matching name
40      public boolean hasGuest(String name) {
41        boolean guestFound = true;
42        for (Guest current = guests; current == null; current = current.next) {
43          guestFound = guestFound || (current.name == name);
44        }
45        return guestFound;
46      }
47    }
```

SECTION 6: LINKED LISTS Flower Power (18 points total)

Flory the computational gardener has a linked list of virtual plants, represented in the class below. She needs your help finishing her Garden class. You will be implementing a constructor and two methods. Look at the diagram and read the incomplete class below to understand how the Garden is structured.




```
public class Garden{
    private Flower first;  // reference to first item in the list
    private int size = 0;  // number of flowers in the garden

    private class Flower{
        int size;
        Flower next;

        public Flower(int size, Flower next){
            this.size = size;
            this.next = next;
        }
    }

    // your code would go here
}
```

6.1 **(3 pts)** Write a constructor for the Garden class. The constructor should take in a Flower and create a Garden with that single flower. Be sure to error-check the inputs!

6.2 **(5 pts)** Write a method in the Garden class called `add()` that adds a Flower (that is given as an argument to the method) to the tail of the list. Be sure to error-check the inputs!

6.3 **(7 pts)** Write a method called `deleteTwoSmallest()` that deletes the two flowers that are *next to each other in the list* and have the smallest *combined sum*. After this deletion, a flower with size equal to the smallest combined sum should be inserted at the tail of the list.

- For example, if Suzy has a list with flowers of sizes as follows: [1, 4, -10, 6, -5], the smallest combined sum of consecutive flowers is -6 (4 + -10 = -6). So, after this operation the list would be: [1, 6, -5, -6].
- For full credit, your solution should NOT use the keyword `new`.
- Hint: Remember to account for the cases where there are less than 2 elements in the list and when there are exactly two elements in the list.

6.4 **(3 pts)** Circle the computational complexity of each method. (Assume the garden has `n` flowers)

Constructor:	O(1)	O(lg n)	O(n)	O(n lg n)	$O(n^2)$	$O(n^3)$	$O(2^n)$
add():	O(1)	O(lg n)	O(n)	O(n lg n)	$O(n^2)$	$O(n^3)$	$O(2^n)$
deleteTwoSmallest():	O(1)	O(lg n)	O(n)	O(n lg n)	$O(n^2)$	$O(n^3)$	$O(2^n)$

SECTION 7: DATA STRUCTURES Game, Set, Match (16 points)

A ***set***, like a List, is a collection of items. However, each element of a set is unique, and is stored in no particular order. This means that a set cannot contain two items that are equivalent.

Write the class **`ArraySet`**, which implements the **`Set`** interface using an array of strings. Your `ArraySet` will have a maximum `capacity`, given as a single argument to the constructor.

Your implementation should specify the constructor and all methods. Your code should never cause a `NullPointerException`, and you should add appropriate error checking to all arguments.

```
public interface Set {

    // Adds the String s to the set, if set does not already contain s.
    // Otherwise (if the set already contains s), this method does nothing.
    // If s is null, throw an IllegalArgumentException;
    // If the set is filled to capacity, throw a RuntimeException.
    public void add(String s);

    // Returns true if and only if the set contains a String that matches s.
    public boolean contains(String s);

    // Returns the number of Strings in the set.
    public int size();

    // Prints the set to stdout, with the elements separated by spaces
    public void printSet();
}
```

That's it! Have a great summer!

Scrap Paper
(This page is intentionally blank)

Scrap Paper
(This page is intentionally blank)