CIS 110 — Introduction to Computer Programming
Summer 2014 — Final

Name: _____

PennKey (e.g., bhusnur4): _____

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 to do so by the proctor**. You will have exactly 115 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 code must be written out in proper Java format, including all curly braces and semicolons.
- Do not separate the pages. If a page becomes loose, reattach 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.
- We wish you the best of luck. Enjoy the rest of your summer!

Scores: [For instructor use only]

Question 0		1 pt
Question 1		17 pts
Question 2		16 pts
Question 3		13 pts
Question 4		13 pts
Total:		**60 pts**

0.) THE EASY ONE (1 point)

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

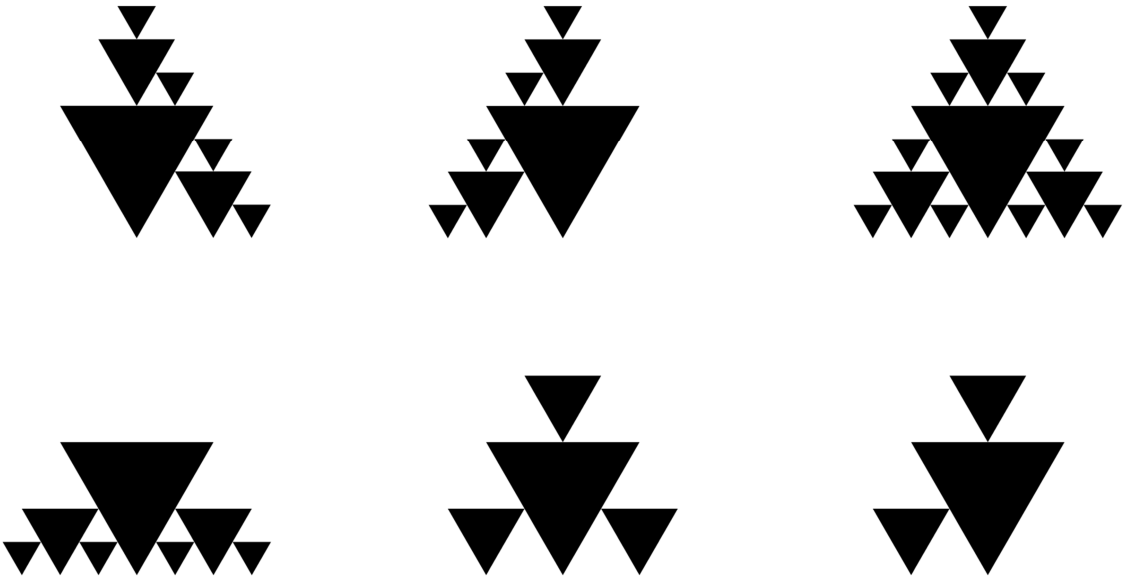
1.) RECURSION (17 points total)

1.1) Bad Sierpinski (2 points)

A student wrote the following code for the Sierpinski triangle assignment. If you look carefully, you will notice a logical flaw which results in a weird shape when they run the program. Which shape will it result in? **Circle the result of the code below**. Assume that `singleTriangle()` draws a single equilateral triangle with the bottom vertex at (x, y).

```
public class Sierpinski {
   public static void sierpinski(int n, double size, double x, double y) {
      if (n == 0) {
         return;
      }
      singleTriangle(size, x, y);
      sierpinski(n - 1, size / 2, x, y + size * Math.sqrt(3) / 2.0);
      sierpinski(n - 1, size / 2, x – size / 2, y);
   }

   public static void main(String[] args){
      sierpinski(3, 0.5, 0.5, 0);
   }
}
```



1.2.) Cruel recursion (10 points total)

This scarily named class is doing something with arrays but has very few comments. The comments that are there are correct. Find out what it is doing.

1) What will be printed when we run `java RecursiveNightmare`? (6 points)

2) In 20 words or fewer, explain what `meanie()` does. `meanie()` is a weird name for a function; what would be a better name for it? (4 points)

```
public class RecursiveNightmare {
    public static double meanie(int[] x) {
        int kruger = x.length;
        if (kruger == 0) {
            return 0;
        }
        if (kruger == 1) {
            return x[0];
        }
        int[] temp = new int[kruger / 2];
        int[] temp2 = new int[kruger - kruger / 2];
        // copy half the array into temp
        for (int i = 0; i < kruger / 2; i++) {
            temp[i] = x[i];
        }
        // copy the second half into temp2
        for (int i = kruger / 2; i < kruger; i++) {
            temp2[i - kruger / 2] = x[i];
        }
        double freddie = temp.length + temp2.length;
        return (meanie(temp) * temp.length
                + meanie(temp2) * temp2.length) / freddie ;
    }

    public static void main(String[] args) {
        int[] abc = {1, 0 , -1, 2, 2, -1, 1, 0};
        double r = meanie(abc);
        System.out.println(r);
    }
}
```

1.3) Walk, trot, Cantor, gallop and recurse! (5 points)

One popular recursive drawing corresponds to a famous mathematical construction by Georg Cantor. The Cantor set is created by repeatedly deleting the open middle third of a set of line segments. **Complete the `cantorize()` and `main()` functions** so that when the `Cantor` class is run with a command-line argument `N`, a representation of the Cantor set is drawn to `N` levels. The command `java Cantor 6` should draw a Cantor set to 6 levels (beginning our numbering from 1), as seen in the picture below. Your code must involve recursion.

As long as the pattern below is produced to any scale, your answer is correct. You may assume that `drawLine(x, y, len)` draws a horizontal line beginning at (`x`, `y`) with length `len`. The exact pixel value of the line length does not matter.

```
public class Cantor {
   public static void drawLine(double x, double y, double len) {
      StdDraw.line(x, y, x + len, y);
   }

   public static void cantorize( /* TODO */                              ) {
   // TODO – complete the body of this method









   }

   public static void main(String[] args) {
   // TODO – complete the body of this method




   }
}
```

2) OOPs :((16 points total)

2.1) True or false? To initialize a variable of type `Object`, you have to call the constructor of `Object`. **(1 point)**

2.2) Pow (5 points)

Here is the code for a class representing complex numbers (from the textbook). We would like to add another method called `pow()` that takes an integer `i` and computes the value of the complex number raised to the `i`th power. **Write the pow() method.**
Remember that *a* to the `i`th power is $a \times a \times a \times \ldots$ `i` times. You can assume that `i` is positive.

```
public class Complex {
   private final double re;
   private final double im;
   public Complex(double real, double imag) {
      re = real;
      im = imag;
   }
   public String toString() { return re + " + " + im + "i"; }
   public double abs() { return Math.sqrt(re * re + im * im); }
   public Complex plus(Complex b) {
      double real = re + b.re;
      double imag = im + b.im;
      return new Complex(real, imag);
   }
   public Complex times(Complex b) {
      double real = re * b.re - im * b.im;
      double imag = re * b.im + im * b.re;
      return new Complex(real, imag);
   }
   // TODO - write the pow() method here

}
```

2.3) Bugs (7 points)

The following attempt has been made at writing a class with some instance variables and methods. It is riddled with bugs. **Correct these bugs.** (Sorry, we are not telling you the number of bugs this time.) Fill your answers in the next page where we have a blank space for each line; you may not need all of the spaces provided. You may assume that the author of the code has given the fields and methods names that reflect their intention.

```
1   public class StudentDemo {
2      public static void main(String[] args){
3         // create a student named Eric Eaton
4         Student s = Student("Eric Eaton", 19, "South Korea");
5         if (s.canVote()) {
6            System.out.println("The student named " + s.name + " can vote");
7         }
8      }
9   }
10
11  class Student {
12     private String name;
13     private String country = "Canada";
14     private int age = 18;
15
16     public Constructor(String n, String ctry, int age) {
17        // initialize name, country and age
18        String country = ctry;
19        name = n;
20        age = age;
21     }
22
23     public static int getVotingAge(String country) {
24        if (country.equals("Singapore")) return 21;
25        else if (country.equals("South Korea")) return 19;
26     }
27
28     public boolean canVote() {
29        if (age >= getVotingAge()) return false;
30        return true;
31     }
32  } // end of Student class
```

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

Line _____ should be replaced by

_____

2.4) JButton (3 points)

Java has many built-in classes. Among them, there is one called `JButton`, which creates a button like those you see on web pages; another is `Dimension`, which is used to represent the width and height of user interface elements such as buttons; yet another is `Color`, which represents the colors that are displayed on screen.

Below are extracts from the public APIs for those classes.

JButton

`JButton()`	Creates a button with no text
`JButton(String text)`	Creates a button with the specified text
`void setPreferredSize(Dimension d)`	Sets the size of the button to `d`
`void setBackground(Color c)`	Sets the background color of the button to `c`

Dimension

`Dimension()`	Creates an instance of `Dimension` representing a height of 0 and a width of 0
`Dimension(int width, int height)`	Creates an instance of Dimension representing specified `height` and `width`

Color

`Color(int red, int green, int blue)`	Creates an instance of `Color` representing a color with specified `red`, `green`, and `blue` components

Suppose we have a `JButton` which has been declared as follows:

```
JButton submitButton;
```

We would like to make `submitButton` refer to a blue button with width 80 and height 40, with the text "Submit" inside it. **Write code that does this,** using the API documentation that we have provided above. (Your code will run in the same scope as the above declaration.)
Recall that blue is represented by 0 red, 0 green and 255 blue.

```
// TODO – write your code here
```

3) SOARTING (13 points total)

3.1) True or false? Mergesort is the fastest way to sort any array, regardless of the data within it. **(1 point)**

3.2) If, on average, it takes 10 seconds to sort an array of length 10,000 using insertion sort, how long will it take to sort an array of length 20,000 using the same algorithm (all other things being equal)? **(2 points)**

3.3) Hors d'œuvre (4 points)

Assume we are using selection sort on the character array

```
char[] name = {'b','e','n','e','d','i','c','t'};
```

(where we sort the characters in alphabetical order). **What is the state of `name` after each step of the selection sort algorithm?** Fill in the table below, using as many rows as necessary to complete the selection sort algorithm.

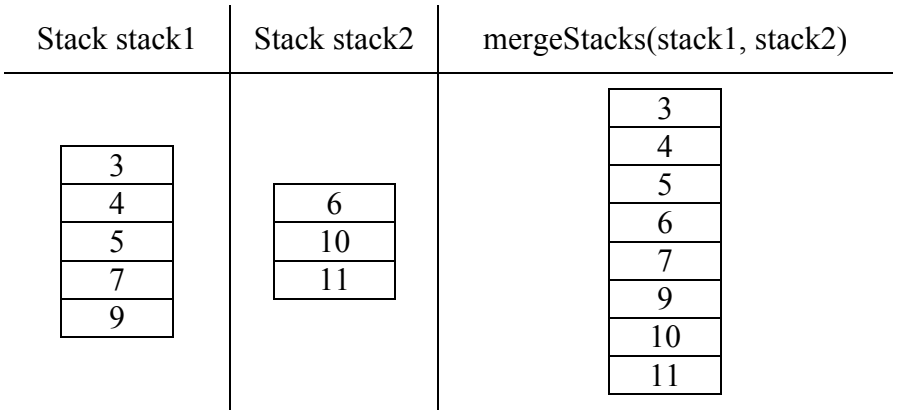
name[0]	name[1]	name[2]	name[3]	name[4]	name[5]	name[6]	name[7]

3.4) Appetizer (6 points)

Assume you have two `Stacks`, both of which contain `int`s in sorted order. `Stack` has the following public API.

`Stack()`	Creates a `Stack`
`void push(int i)`	Push `i` to the top of the `Stack`
`int pop()`	Pop the element at the top of the `Stack` and return it
`int peek()`	Return the element at the top of the `Stack`, without popping it
`boolean isEmpty()`	Returns true if the `Stack` is empty

Write a public static method `mergeStacks()` that takes two sorted `Stacks`, and returns a new `Stack` where the elements of the two `Stacks` are merged. The merging is to be done in the same manner as you have seen with the mergesort of arrays, as illustrated below.




```
public static Stack mergeStacks(Stack stack1, Stack stack2) {
    // TODO – complete the body of this method




}
```

4) LINKED LISTS (13 points)

4.1) Main course (10 points)

Assume that we have defined the following classes.

```
public class Node {
   public int item;
   public Node next;
}

public class LinkedList {
   public Node first;
}
```

In the `LinkedList` class, **write a public void method, `delAlternate()`**, that removes every other element in the list. (If we give `first` the index 0, then only the elements with even indices should remain in the list.) Be careful to cover all cases!

```
public void delAlternate() {
   // TODO – complete the body of this method









}
```

4.2) Dessert (3 points)

Assume that we have defined the following classes.

```
public class Student {
   private String name;
   private String college;
}

public class Course {
   private String name;
   private String department;
   private String semester;
}
```

You want to add another field to the `Student` class. It is called `courses` and it represents all the courses that the `Student` has taken so far.

The most common thing that a `Student` does is take courses. Courses can be added and dropped and `Students` tend to do both those things a lot, so the data structure needs to be optimized for that.

Would you pick an array or linked list representation for `courses`? Explain why. Given your choice, will it be simpler (in terms of the number of operations needed) to add a course, or will it be simpler to drop a particular course in `courses`? Explain why.

You need not write any code: this is a conceptual question.

Extra page

This page intentionally left blank

Extra page

This page intentionally left blank