COS 126	Princeton University	Fall 2022

Written Exam 1

This exam has 10 questions worth a total of 50 points. You have 60 minutes.

Instructions. This exam is preprocessed by computer. Write neatly, legibly, and darkly. Put all answers (and nothing else) inside the designated spaces. Fill in bubbles and checkboxes completely: ● and ■. To change an answer, erase it completely and redo.

Resources. The exam is closed book, except that you are allowed to use a one page reference sheet (8.5-by-11 paper, one side, in your own handwriting). No electronic devices are permitted.

Honor Code. This exam is governed by Princeton's Honor Code. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

Please complete the following information now.

Name:

NetID:

Exam room: ◯ McCosh 10 ◯ McCosh 50 ◯ McCosh 62 ◯ Other

Precept:

P01	P02	P03	P04	P05	P06	P07	P08	P08A
◯	◯	◯	◯	◯	◯	◯	◯	◯

P10	P11	P12	P12A	P13	P14	P15	ISC
◯	◯	◯	◯	◯	◯	◯	◯

"I pledge my honor that I will not violate the Honor Code during this examination."

Signature

1. **Initialization. (1 point)**

   On the front of this exam, in the designated spaces:

   - Write your name.
   - Write your NetID (not email alias).
   - Mark the room in which you are taking the exam.
   - Mark your precept number.
   - Write and sign the Honor Code pledge.

2. **Java expressions. (7 points)**

Assume that the variables `x`, `y`, and `z` have been declared and initialized as follows:

```
int x = 2;
int y = 3;
int z = 4;
```

Evaluate each Java expression on the left.

For each Java expression on the left, write the letter of the best-matching value on the right. You may use each letter once, more than once, or not at all.

[J]	`1 + 2`	**A.** `false`
[ ]	`Math.sqrt(2.0 * x)`	**B.** `true`
[ ]	`0.0 + y / x * x * 1.0`	**C.** `0`
[ ]	`Math.max(x, Math.min(y, z)) - x`	**D.** `0.0`
[ ]	`(z < x) && (z < y) || (x <= y <= z)`	**E.** `0.75`
[ ]	`!((z < x) && !!(x != y) && !(x >= x))`	**F.** `1`
[ ]	`Double.parseDouble(Integer.parseInt("2"))`	**G.** `1.0`
[ ]	`y / (z % x)`	**H.** `2`
		I. `2.0`
		J. `3`
		K. `3.0`
		L. *compile-time error or run-time exception*

3. **Properties of types, variables, and expressions. (5 points)**

Which of the following are properties of types, variables, and expressions in Java?

Mark each statement as either true or false by filling in the appropriate bubble.

true	*false*	
◯	◯	Every variable has a type (such as `int`, `double`, or `String`) that is known at compile time.
◯	◯	The result of applying one of the arithmetic operators (`+`, `-`, `*`, or `/`) to two `double` operands always evaluates to a value of type `double` (and never produces a run-time exception).
◯	◯	If you attempt to use a local variable of type `int` in an expression before that variable has been assigned a value, Java will substitute the value `0`.
◯	◯	If a variable is declared and initialized in the body of a `for` loop, that variable cannot be accessed outside that loop.
◯	◯	If you name a variable with all uppercase letters and initialize it to some value, attempting to subsequently change its value would lead to a compile-time error.

4. **Type conversion and random numbers. (5 points)**

Let `n` be a variable of type `int` containing a positive integer. Each of the following expressions is intended to evaluate to a "random" *even* integer between 0 (inclusive) and $2n$ (exclusive). For example, if n is 5, each expression should evaluate to `0`, `2`, `4`, `6`, or `8`, each with approximately 20% likelihood.

Which of the following expressions work as intended? Mark all that apply.

Recall that `Math.random()` *returns a* `double` *between 0.0 (inclusive) and 1.0 (exclusive).*

☐ `2 * (int) (n * Math.random())`

☐ `2 * (int) n * Math.random()`

☐ `(int) (2 * n * Math.random())`

☐ `(int) (2 * n * Math.random()) / 2 * 2`

☐ `(int) (2 * n * Math.random()) * 2 / 2`

5. **Loops, debugging, and tracing. (6 points)**

What is the value of the variable `count` immediately after executing each of the following (not necessarily well-engineered) code fragments?

For each code fragment on the left, write the letter of the best-matching value on the right. You may use each letter once, more than once, or not at all.

[J]

```
int n = 10;
int count = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        count++;
```

[ ]

```
int n = 10;
int count = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; i < n; i++)
        count++;
```

[ ]

```
int n = 10;
int count = 0;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n && j != i; j++)
        count++;
```

[ ]

```
int n = 10;
int count = 0;
int i = 0;
int j = 0;
while (i < n) {
    i++;
    while (j < n) {
       j++;
       count++;
    }
 }
```

A. 0

B. 1

C. 2

D. 10

E. 11

F. 20

G. 22

H. 45

I. 90

J. 100

K. 110

L. 121

M. *infinite loop*

6. **Properties of arrays. (6 points)**

Which of the following are properties of arrays in Java?

Mark each statement as either true or false by filling in the appropriate bubble.

true	*false*	
◯	◯	After you create and initialize an `int[]` array, you cannot change its length.
◯	◯	If `a[]` is an array of type `char[]` and length 10, then `a[0.0]` is a valid expression that gives its first element.
◯	◯	It is possible to declare, create, and initialize a `String[]` array without using the keyword `new`.
◯	◯	If `a[]` and `b[]` are two different arrays of the same type and length, then the expression `(a == b)` evaluates to `true` if the corresponding array elements are equal, and `false` otherwise.
◯	◯	The elements of an array of type `int[]` are stored contiguously in the computer's memory (i.e., in consecutive memory locations).
◯	◯	It is possible to create and initialize a two-dimensional array `a[][]` of type `double[][]` such that `(a[0].length != a[1].length)`.

7. **Standard input, standard output, and the command line. (5 points)**

Consider the following Java program:

```
public class Mystery {
    public static void main(String[] args) {
        int a = StdIn.readInt();
        int b = StdIn.readInt();
        StdOut.printf("%d %d\n", b, a + b);
    }
}
```

Suppose that the file `input.txt` contains the two integers `4` and `11`, separated by whitespace. What output will appear in the terminal window after each of the command (or commands) on the left are run?

For each command (or commands) on the left, write the letter of the best-matching output on the right. You may use each letter once, more than once, or not at all. Assume that, for the purposes of this question, the `java` *command is equivalent to* `java-introcs`*.*

☐	`java Mystery 4 11`	**A.** `4 11`
☐	`java Mystery < input.txt`	**B.** `11 15`
☐	`java Mystery < input.txt > output.txt`	**C.** `15 26`
☐	`java Mystery < input.txt > output.txt` `java Mystery < output.txt`	**D.** `26 41`
☐	`java Mystery < input.txt | java Mystery | java Mystery | java Mystery`	**E.** `41 67`
		F. `67 108`
		G. *nothing*
		H. *run-time exception*

8. **Standard drawing, loops, and conditionals. (5 points)**

Consider the following code fragment, which draws an n-by-n grid of filled circles. Recall that `StdDraw.filledCircle(x, y, r)` draws a filled circle of radius r, centered at (x, y), in the current pen color.

```
// lower-left  endpoint = (0, 0)
// upper-right endpoint = (n, n)
StdDraw.setXscale(0, n);
StdDraw.setYscale(0, n);

// draw the n-by-n grid of filled circles
for (int y = 0; y < n; y++) {
    for (int x = 0; x < n; x++) {
        if ((x + y) % 3 == 0) StdDraw.setPenColor(StdDraw.RED);
        if ((x + y) % 3 == 1) StdDraw.setPenColor(StdDraw.GREEN);
        if ((x + y) % 3 == 2) StdDraw.setPenColor(StdDraw.BLUE);
        StdDraw.filledCircle(x + 0.5, y + 0.5, 0.5);
    }
}
```

Which of the following properties are true for *every* value of n between 1 and 100?

Mark all that apply.

■ It draws an n-by-n grid of filled circles, with each circle colored red, green, or blue.

☐ The lower-leftmost circle is red.

☐ The upper-leftmost and lower-rightmost circles are the same color.

☐ The upper-leftmost circle is drawn last.

☐ No two circles of the same color touch.

☐ If the order of the `x` and `y` loops are swapped, the code fragment produces exactly the same final drawing (but the circles may be drawn in a different order).

9. **Incremental design and n-body assignments. (5 points)**

   Recall the following steps in the n-body programming assignment:

   1. Parse the command-line arguments T and Δt.
   2. Read the universe from standard input (i.e., number of bodies, radius of universe, initial positions and velocities, masses, image filenames).
   3. Initialize standard drawing (e.g., setting the x- and y-scales).
   4. Play music on standard audio.
   5. Simulate the universe:
      A. Calculate net forces.
      B. Update velocities and positions.
      C. Draw the universe to standard drawing.
   6. Print the universe to standard output.

   Suppose that, starting from a correct solution implementing these steps, you swap the order of two of the above steps (and make no other changes). For which would the program still compile and produce identical output on both standard drawing and standard output?

   Mark all that apply.

   ☐ Swap steps 1 and 2.

   ☐ Swap steps 2 and 3.

   ☐ Swap steps 3 and 4.

   ☐ Swap steps 5A and 5B.

   ☐ Swap steps 5B and 5C.

10. **Functions, arrays, and pass-by-value. (5 points)**

Consider the following two functions, which purport to negate the values in an integer array:

```
public static void negate1(int[] a) {
    for (int i = 0; i < a.length; i++)
       a[i] = -a[i];
}

public static int[] negate2(int[] a) {
    int[] b = new int[a.length];
    for (int i = 0; i < a.length; i++)
        b[i] = -a[i];
    a = b;
    return b;
}
```

Suppose that an integer array `a[]` in `main()` is declared and initialized to contain the three integers `[1, 2, 3]`. What will be the contents in the array referenced by `a[]` after executing each of the following statements? Answer each part independently.

For each statement on the left, write the letter of the best-matching description on the right. You may use each letter once, more than once, or not at all.

☐	`negate1(a);`	**A.** `[1, 2, 3]`
☐	`negate2(a);`	**B.** `[-1, -2, -3]`
☐	`a = negate2(a);`	**C.** `[0, 0, 0]`
☐	`negate1(negate2(a));`	**D.** *compile-time error or run-time exception*
☐	`a = negate2(negate2(negate2(a)));`	