COS 126	Princeton University	Fall 2024

Written Exam 1 Solutions

This exam has 10 questions worth a total of 100 points. You have 80 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 are posted is a violation of the Honor Code.

Please complete the following information now.

Name: Ada Lovelace

NetID: alovelace

Exam room: ● McCosh 50 ○ McCosh 66 ○ Robertson 100 ○ Other

Precept:

P01	P01A	P02	P03	P03A	P03B	P04	P04A	P05	P05A
○	○	○	○	○	○	○	○	○	○

P06	P10	P10A	P11	P12	P13	P14	P14A	P15
○	○	○	●	○	○	○	○	○

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

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

Ada Lovelace

Signature

1. **Initialization. (2 points)**

   In the designated spaces on the front of this exam,

   - *Write* your name.
   - *Write* your Princeton NetID (6–8 alphanumeric characters).
   - *Fill in* the bubble corresponding to the room in which you are taking this exam.
   - *Fill in* the bubble corresponding to your precept.
   - *Write* and *sign* the Honor Code pledge.

2. **Java expressions. (8 points)**

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

```
int x = 1;
int y = 2;
int z = 6;
```

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

I	`1 + 1`	**A.** `false`
P	`x / (y / z)`	**B.** `true`
D	`z % x % y`	**C.** `-1`
C	`y - x  *  y - x`	**D.** `0`
A	`!!!(x <= y)`	**E.** `1`
F	`Math.sqrt(x + y + z) / (x + y)`	**F.** `1.0`
P	`Integer.parseInt(x + "0.75")`	**G.** `1.5`
E	`(int) Math.min(1.75, Math.max(x, y))`	**H.** `1.75`
O	`(double) z / (y - 2)`	**I.** `2`
		J. `2.0`
		K. `3`
		L. `3.0`
		M. `6`
		N. `10.75`
		O. *positive infinity*
		P. *compile-time error or run-time exception*

3. **Programming terminology. (10 points)**

For each programming term on the left, write the letter of the best-matching description from the right. Use each letter exactly once.

[G] Recursion	**A.** Converts a Java program into machine-language code suitable for execution by a computer.
[A] Compilation	**B.** Two functions with the same name, but different ordered lists of parameter types.
[K] Data type	**C.** A storage location for a data-type value.
[C] Variable	**D.** Specifies method headers and behavior of a library.
[F] Pass by value	**E.** The region of a program where a particular variable is accessible.
[E] Scope	**F.** Copies parameter values and assigns to argument variables.
[B] Overloading	**G.** A function that calls itself.
[D] API	**H.** Makes the output of one program become the input of another program.
[H] Piping	**I.** Uses a file as the input to a program (instead of from keyboard/terminal).
[I] Input redirection	**J.** Sends the output of a program to a file (instead of to display/terminal).
[J] Output redirection	**K.** A set of values and operations on those values.

4. **Flow control. (10 points)**

(a) Consider the following code fragment:

```
boolean a = true;
boolean b = true;
while ((a && b) || !(a || b)) {
    // true or false with 50/50 chance
    a = (Math.random() < 0.5);
    b = (Math.random() < 0.5);
}
// END OF LOOP
```

What can you conclude about `a` *and* `b` *at the end of the loop? Fill in the best answer.*

- ○ The values of `a` and `b` are both `true`.
- ○ The values of `a` and `b` are both `false`.
- ○ The values of `a` and `b` are equal.
- ● The values of `a` and `b` are different.
- ○ The loop will never terminate.

(b) Consider the following code fragment:

```
int n = 100;
for (int i = 0; i < n; i++) {
    for (int j = n-1; j >= 0 && j != i; j--) {
        StdOut.println(i + "-" + j);
    }
}
```

Which of the following will get printed? Fill in all checkboxes that apply.

- ■ `0-99`
- □ `0-0`
- ■ `10-20`
- □ `12-6`
- □ `50-50`
- □ `99-1`

(c) Consider the following function `f()`:

```
public static String f(int a) {
    if (a < 0) {
        return "A";
    }
    else if (a >= 5) {
        if (a <= 5) {
            return "B";
        }
        return "C";
    }
    else if (Math.abs(a) == 5) {
        return "D";
    }
    if (a >= 3) {
        return "E";
    }
    if (a >= -1) {
        return "F";
    }
    return "G";
}
```

Which of the following strings could `f(a)` *possibly return (for some value of* `a`*)? Fill in all checkboxes that apply.*

■	■	■	□	■	■	□	□
"A"	"B"	"C"	"D"	"E"	"F"	"G"	"H"

5. **Standard input, loops, and conditionals. (12 points)**

Consider the Java program `Test.java` and the text file `input.txt`:

```
public class Test {
    public static void main(String[] args) {
        int result = -1;
        while (!StdIn.isEmpty()) {
            int a = StdIn.readInt();

            // SUBSTITUTE EACH CODE
            // FRAGMENT HERE INDEPENDENTLY

        }
        StdOut.println(result);
    }
}
```

```
% more input.txt
1 2 2 3 1
6 3 5 3 4
```

Substitute each code fragment below into the designated place in the program above. What is the result of executing the program with `java-introcs Test < input.txt`?

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

H

```
if (a > result) {
    result = a;
}
```

A

```
if (a < result) {
    result = a;
}
```

F

```
result = a;
```

I

```
if (a > result) {
    result = StdIn.readInt();
}
```

A. `-1`

B. `0`

C. `1`

D. `2`

E. `3`

F. `4`

G. `5`

H. `6`

I. *run-time exception*

6. **Properties of functions. (12 points)**

Which of the following are properties of *functions* in Java?

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

true	*false*	
●	○	In Java, a *function* is implemented as a `static` method.
●	○	A `void` function *may* contain a `return` statement.
○	●	A function that produces a side effect *must* be declared as `void`.
○	●	If two functions defined in the same class have different return types, they *must* have different names.
○	●	If a function is defined using the `private` access modifier, it *cannot* be called from a function that uses the `public` access modifier.
○	●	A local variable declared within the body of a `public` function *can* be accessed by name from any other `public` function in the class.
●	○	If you pass a value of type `int` to a function that takes an argument of type `String`, that will produce a *compile-time error*.

7. **Arrays and loops. (12 points)**

Each of the code fragments on the left is an attempt to *reverse* the elements within a `String` array `a[]` of length `n`, but not all of them succeed in doing so.

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

E

```
for (int i = 0; i < n / 2; i++) {
    String temp = a[i];
    a[i] = a[n-i];
    a[n-i] = temp;
}
```

A

```
for (int i = 0; i < n / 2; i++) {
    String temp = a[i];
    a[i] = a[n-i-1];
    a[n-i-1] = temp;
}
```

B

```
for (int i = 0; i < n; i++) {
    String temp = a[i];
    a[i] = a[n-i-1];
    a[n-i-1] = temp;
}
```

A

```
int i = 0;
int j = n - 1;
while (i < j) {
    String temp = a[i];
    a[i] = a[j];
    a[j] = temp;
    i++;
    j--;
}
```

A. *reverses the elements in* `a[]`

B. *leaves the elements of* `a[]` *unchanged*

C. *shuffles the elements of* `a[]`

D. *infinite loop*

E. `ArrayIndexOutOfBoundsException`

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

Consider the following Java functions:

```
public static int increment1(int x) {
    x++;
    return x;
}

public static int[] increment2(int[] b) {
    for (int i = 0; i < b.length; i++)
        b[i] = b[i] + 1;
    return b;
}

public static void increment3(int[] b) {
    for (int i = 0; i < b.length; i++)
        increment1(increment1(b[i]));
}

public static void increment4(int[] b) {
    int[] c = b;
    for (int i = 0; i < c.length; i++) {
        b[i] = c[i] + 1;
        c[i] = b[i] + 1;
    }
}
```

Suppose that an integer array `a[]` in `main()` contains the three integers `[1, 2, 6]`. What will be the contents of `a[]` after executing each of the following statements?

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

F	`a++;`	**A.** `[1, 2, 6]`
B	`increment2(a);`	**B.** `[2, 3, 7]`
A	`increment3(a);`	**C.** `[3, 4, 8]`
C	`increment4(a);`	**D.** `[4, 5, 9]`
C	`increment2(increment2(a));`	**E.** `[1, 1, 1]`
		F. Produces a *compile-time error* or a *run-time exception*

9. **Recursive graphics. (12 points)**

Consider the recursive function `draw()` that draws an H-tree of order n:

```
// draw an H-tree of order n, centered at (x, y), of specified size
public static void draw(int n, double size, double x, double y) {
    if (n == 0) return;                                              1
    drawOneH(size, x, y);                          // draw an H      2
    draw(n-1, size/2, x - size/2, y - size/2);  // lower left      3
    draw(n-1, size/2, x - size/2, y + size/2);  // upper left      4
    draw(n-1, size/2, x + size/2, y - size/2);  // lower right     5
    draw(n-1, size/2, x + size/2, y + size/2);  // upper right     6
}
```

Suppose that we reorder the 6 statements in `draw()` and pause execution immediately after the 30th call to `drawOneH()`. Each call to `drawOneH()` draws one H, centered at (x, y), of the specified size.

For each of the following orderings of the 6 statements, write the letter corresponding the picture that results from calling `draw(4, 0.5, 0.5, 0.5)` *and pausing execution as described above. You may use each letter once, more than once, or not at all.*

D — 1 2 3 4 5 6

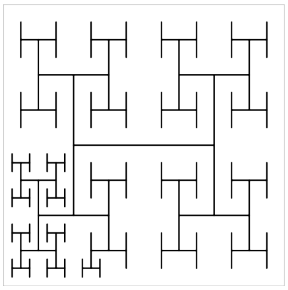
C — 1 2 4 6 3 5

B — 2 1 3 4 5 6

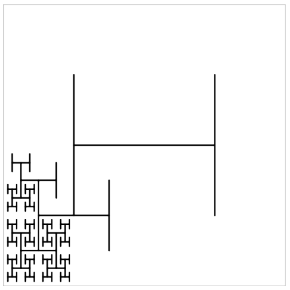
E — 1 3 4 5 6 2

F — 3 4 1 2 5 6

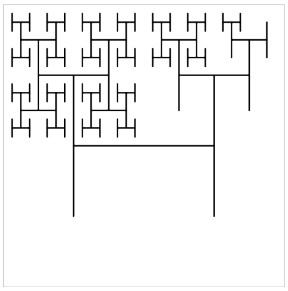
A.



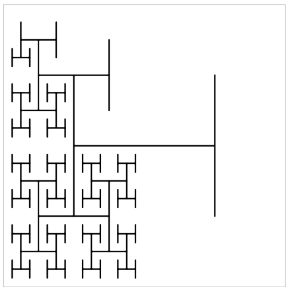
B.



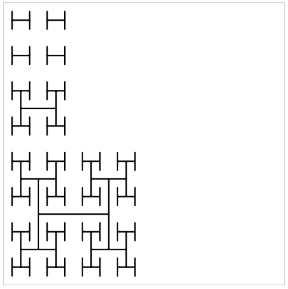
C.



D.



E.



F. `StackOverflowError`

10. **Performance. (12 points)**

Determine the *order-of-growth of the running time* of each of the following code fragments as a function of n.

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

F

```
public static int f1(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            count++;
    return count;
}
```

E

```
public static int f2(int n) {
    int count = 0;
    for (int i = n; i >= 1; i = i/2)
        for (int j = n; j >= 1; j--)
            count++;
    return count;
}
```

B

```
public static int f3(int n) {
    if (n == 0) return 0;
    return 1 + f3(n / 2);
}
```

H

```
public static int f4(int n) {
    int count = 0;
    for (int i = 0; i < n; i++)
        count++;
    for (int i = 0; i < f1(n); i++)
        count += f1(n);
    return count;
}
```

A. $\Theta(1)$ *constant*

B. $\Theta(\log n)$ *logarithmic*

C. $\Theta(\sqrt{n})$ *square root*

D. $\Theta(n)$ *linear*

E. $\Theta(n \log n)$ *linearithmic*

F. $\Theta(n^2)$ *quadratic*

G. $\Theta(n^3)$ *cubic*

H. $\Theta(n^4)$ *quartic*

I. $\Theta(2^n)$ *exponential*

J. *infinite loop*