COS 126	Princeton University	Fall 2025

Written Exam 1

This exam has 9 questions worth a total of 68 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 completely, like this: ● . To change an answer, erase it completely and redo. Do not remove or unstaple any pages. You may use the last page for scratch work, as well as margins, provided it does not interfere with boxes and bubbles for answers. Additional scratch paper is available upon request.

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:

NetID:

Exam room: ◯ McCosh 46 ◯ McCosh 50 ◯ Other

Precept:

P01	P01A	P02	P03	P03A	P04	P04A	P05	P05A
◯	◯	◯	◯	◯	◯	◯	◯	◯

P06	P10	P10A	P10B	P11	P11A	P12
◯	◯	◯	◯	◯	◯	◯

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

Signature

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

   In the designated spaces on the front of this exam,

   - *Print* your name.
   - *Print* your Princeton NetID (6–8 characters, like "`cs0126`").
   - *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 (10 points)**

For each Java expression below, write the **letter** corresponding to its value from the list of possible answers. Answers may be used more than once; not all answers will be used.

Answer	Expression
	`(int)(7.8 - 2.3)`
	`15 % 4 + 2`
	`20 / 6`
	`!(false || (4 < 2))`
	`Integer.parseInt("6.5")`
	`2 + 12 / 5.0`
	`Math.sqrt(25) / 2`
	`Math.pow(2, 3) - 3.6`
	`"ab" + 1 + 2`
	`4 <= (2 * 2.0)`

Possible Answers:

A. `2`
B. `3`
C. `4`
D. `5`
E. `6`
F. `7`
G. `ab12`
H. `ab3`
I. `3ab`
J. `12ab`
K. `true`
L. `false`
M. *Error*
N. `2.5`
O. `2.8`
P. `4.4`
Q. `4.0`
R. `5.0`
S. `5.5`

3. **Java (10 points)**

For each of the following statements, fill in the bubble corresponding to *true* or *false*.

true	*false*	
◯	◯	An `if` statement must always have an accompanying `else` clause.
◯	◯	The operator `==` can be used to compare two `int` values.
◯	◯	A method can have multiple `return` statements.
◯	◯	The keyword `public` means that a method can be called from any other class.
◯	◯	A `while` loop always executes its body at least once.
◯	◯	The last element in an array `int[] a` is `a[a.length]`.
◯	◯	Variables declared inside a block (such as an `if` statement) are always accessible outside that block.
◯	◯	Elements of an array `temp` created with the following command are automatically initialized to 0: `int[] temp = new int[5];`
◯	◯	The condition of a `for` loop is checked at least once.
◯	◯	Two methods in the same class may have the same name as long as their parameter lists differ.

4. **Loops and tracing (8 points)**

What is the value of the variable `count` immediately after executing each of the following code fragments? Write the **letter** corresponding to the correct answer from the list of possible answers. Assume that each code fragment is independent (i.e., appears in a separate program). You may use each letter once, more than once, or not at all.

Answer **Code Fragment**

☐
```
int n = 4;
int count = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j <= i; j++) {
        count = count + 1;
    }
}
```

☐
```
int i = 20;
int count = 0;
while (i > 1) {
    count = count + 1;
    i = i / 2;
}
```

☐
```
int i = 0;
int count = 0;
while (i < 10) {
    count = count + 1;
    if (i % 2 == 0) {
        i = i + 1;
    } else {
        i = i - 1;
    }
}
```

☐
```
int i = 10;
int count = 0;
while (i > 0) {
    count = count + 1;
    i = i - 4;
}
```

Possible Answers:

A. `0`

B. `3`

C. `4`

D. `5`

E. `6`

F. `7`

G. `8`

H. `9`

I. `10`

J. `12`

K. *Infinite loop*

5. **Input, Output, Redirection, and Piping (8 points)**

Consider the following three Java programs. Assume `StdIn` and `StdOut` libraries are available both at compilation and runtime.

Program `Sum.java`:

```
public class Sum {
    public static void main(String[] args) {
        int total = 0;
        while (!StdIn.isEmpty()) {
            int x = StdIn.readInt();
            total += x;
        }
        StdOut.println(total);
    }
}
```

Program `Repeat.java`:

```
public class Repeat {
    public static void main(String[] args) {
        while (!StdIn.isEmpty()) {
            String s = StdIn.readString();
            StdOut.println(s + s);
        }
    }
}
```

Program `Count.java`:

```
public class Count {
    public static void main(String[] args) {
        int count = 0;
        while (!StdIn.isEmpty()) {
            StdIn.readString();
            count++;
        }
        StdOut.println(count);
    }
}
```

(See next page.)

Two files are available:

File `words.txt`:

```
hi
bye
```

File `data.txt`:

```
3
7
```

For each of the following statements about running these programs, fill in the bubble corresponding to *true* or *false*. Assume that running with `java` behaves the same as with `java-introcs`.

true	*false*	
◯	◯	`java Repeat < words.txt > out.txt` creates a file containing `hihi` and `byebye`.
◯	◯	`java Repeat words.txt | java Count` prints `2`.
◯	◯	`java Sum < data.txt` prints `10`.
◯	◯	`java Sum "120 6"` returns `126`.
◯	◯	`java Repeat < words.txt | java Count` prints `2`.
◯	◯	`java Sum < words.txt` prints `0`.
◯	◯	`java Repeat < data.txt | Sum` prints `110`.
◯	◯	`java Count < words.txt | java Sum | java Repeat` prints `22`.

6. **Functions (6 points)**

Answer each of the following questions about Java methods. Assume that each code fragment is independent (i.e., appears in a separate program). Write your answer in the box provided.

(a)

```
public static void mystery(int[] arr) {
    arr = new int[3];
    arr[0] = 42;
}
public static void main(String[] args) {
    int[] a = {1, 2, 3};
    mystery(a);
    StdOut.println(a[0]);
}
```

What will be printed by `main`?

(b)

```
public static void mystery(int x) {
    x = x + 5;
}
public static void main(String[] args) {
    int a = 10;
    mystery(a);
    StdOut.println(a);
}
```

What will be printed by `main`?

(c)

```
public static int f(int n) {
    return n + g(n);
}
public static int g(int n) {
    return n * 2;
}
```

What is the value of the call `g(f(2))`?

7. **Arrays (8 points)**

Determine whether each of the following lines of code *compiles* or *does not compile*. Assume that each line of code is independent (i.e., appears in a separate program).

compiles	*does not compile*	
◯	◯	`double[] d = new double[3]; d[0] = 2;`
◯	◯	`int[] a = new int[3]; a[0] = 2.5;`
◯	◯	`int[] a = new int[3]; double i = 1.0; int v = a[i];`
◯	◯	`int[] a = {1,2,3}; int i; a[i] = 0;`
◯	◯	`int[] a = new int[3]; int[] b = a;`
◯	◯	`double[] d = new double[3]; int k = d[0];`
◯	◯	`int[] a; a = new int[3];`
◯	◯	`int[] a = new int[3]; int b = a.length();`

8. **Recursion (10 points)**

Consider the following recursive function:

```
public static String f(int n) {
    if (n <= 0) return "0";
    if (n == 1) return "1";
    return f(n / 2) + ":" + n + ":" + f(n - n / 2);
}
```

A *token* is any piece of text between two colons, or between the beginning or end of the string and the closest colon. For example, the string `"2:3:4"` has three tokens: `"2"`, `"3"`, and `"4"`.

(a) Write the exact output for the following calls:

`f(2)`	`1:2:1`
`f(3)`	
`f(4)`	

(b) For each of the following statements about the function `f`, fill in the bubble corresponding to *true* or *false*.

true	*false*	
◯	◯	For every $n \geq 1$, `f(n)` contains exactly n tokens equal to `"1"`.
◯	◯	In `f(126)`, the number of colons is 250.
◯	◯	In `f(65)`, the token `"3"` appears.
◯	◯	While computing `f(21)`, the function `f(5)` is called exactly 2 times.

9. **Performance (6 points)**

Determine the order-of-growth of running time relative to input n, a positive integer. For each code fragment below, write in the corresponding box to its left the **letter** of the best-matching term from the right column. Assume that each code fragment is independent (i.e., appears in a separate program). Answers may be used once, more than once, or not at all.

Answer **Code Fragment**

☐
```
int[] a = new int[n];
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += a[i];
}
```

☐
```
while (n > 1) {
    n = n / 2;
}
```

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

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

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

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

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

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

F. $\Theta(n^2 \log n)$ *quadrithmic*

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

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

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

J. *infinite loop*

This page is intentionally left blank. You may use it for scratch work.