COS126 **Written Exam 1** **Fall** '23

Instructions. This exam has nine (9) questions worth a total of one hundred (100) points. You have eighty (80) minutes.

This exam is preprocessed by computer. Write neatly, legibly, and darkly. If you use a pencil, use extra care to write darkly. Fill in bubbles and checkboxes completely: ⚫ and ⬛ (not ✔ or ✘). Place only your answer inside a box, although you may show work outside a box.

To change an answer, erase it completely and redo.

Resources. The exam is closed book, except that you are allowed to use a single one-sided reference sheet (8.5-by-11 paper, one-sided, 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.

NAME:

NETID

PRECEPT

P01	P02	P02A	P03	P04	P05	P07	P08	P08A
◯	◯	◯	◯	◯	◯	◯	◯	◯
P10	P11	P12	P12A	P13	P14	P15	P16	P16A
◯	◯	◯	◯	◯	◯	◯	◯	◯

EXAM ROOM McCosh 50 ◯ McCosh 10 ◯ McCosh 62 ◯ McCosh 66 ◯ Other _____

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

_____

Signature

Question 1	**Java Data Values and Types**	**12** points

Write the value and type of each of the following expressions. To express a value, write a Java literal of the appropriate form, such as **`0`** (for an **`int`**), **`3.14`** (for a **`double`**), **`false`** (for a **`boolean`**), **`"tiger"`** (for a **`String`**), **`'a'`** (for a **`char`**). If the expression does not compile or causes a runtime exception, place an **`X`** in both boxes.

Expression	Value	Type
`(double)(1 / 3 + 1.0)`		
`Double.parseDouble("+" + 1 + "E+1")`		
`Math.max(true, false)`		
`true || ((3.14 / 1.242) > 2.00012)`		
`(0.5 * (10 / 4))`		
`(int) (12 * 0.2)`		
`(1 > 0) || (1.0 <= 2.0 <= 3.0)`		
`Double.parseDouble("2") + Integer.parseInt("2")`		

Question 2	**Java**	**8** points

Which of the following are features of Java? Fill in the circle corresponding to *true* or *false*:

	true	*false*	
1	◯	◯	Reserved words in Java may never be used as variable names.
2	◯	◯	The scope of a function's parameter variables is limited to that function's body.
3	◯	◯	A static function may never produce a side effect.
4	◯	◯	All static functions must explicitly invoke a `return` statement.
5	◯	◯	Every Java variable must have a declared type.
6	◯	◯	A library is a module whose functions are primarily intended for use by clients, where a library provides an API (application programming interface).
7	◯	◯	All local variables in a static function must be declared at the beginning of the function.
8	◯	◯	All static functions defined in a single Java class must have unique names.

Question 3	**Conditionals**	**12** points

1. Consider the following code snippet and table. For each combination of the **boolean** variables **a**, **b**, and **c** , what is the value for **result** after the code executes. Fill in the circle for **true** or **false**. The first row is filled in for you.

```
boolean result = false;
if (a)
   result = (b || c);
else if (b)
   result = c;
else
   result = b;
```

a	b	c	result
false	false	false	◯ true ● false
false	false	true	◯ true ◯ false
false	true	false	◯ true ◯ false
false	true	true	◯ true ◯ false
true	false	false	◯ true ◯ false
true	false	true	◯ true ◯ false
true	true	false	◯ true ◯ false
true	true	true	◯ true ◯ false

2. Consider the following code snippet and the boolean variables **a**, **b**, and **result**. After the code executes, which of the following statement(s) are correct? Fill in the circle(s) corresponding to the correct statement(s), if any. Incorrect selections will reduce partial credit.

```
if ((!a && b) || (a && !b))
    result = true;
 else
    result = false;
```

◯ Will always assign **result = true** if at least one of **a** and/or **b** is **true**.

◯ Will only assign **result = true** if both **a** and **b** are **true**.

◯ Will always assign **result = true** if **a** and **b** have different values.

◯ Will always assign **result = false** if a is **true**.

◯ Will only assign **result = false** if **a** and **b** have the same value.

◯ Will never assign **result = false**.

◯ I need to know the values of the two **boolean** variables **a** and **b** to determine whether the statements above are correct.

Question 4	**Loops**	**12** points

For each loop on the left, write the letter (in the box) of the matching output on the right when `N == 4`. You may use each letter once or more than once. (Whitespace in output is for readability purposes only.)

1.
```
for (int i = 0; i < N; i++) {
  for (int j = i; j < N; j++)
    System.out.print((N - i + j) % 2);
  System.out.println();
}
```
☐

2.
```
for (int i = 0; i < N; i++) {
  for (int j = N; j > i; j--)
    System.out.print((N - i + j) % 2);
  System.out.println();
}
```
☐

3.
```
for (int i = 0; i < N; i++) {
  for (int j = 0; j < N; j++)
    System.out.print((N - i + j) % 2);
  System.out.println();
}
```
☐

4.
```
for (int i = 0; i < N; i++) {
  for (int j = 1; j <= i + 1; j++)
    System.out.print((N - i + j) % 2);
  System.out.println();
}
```
☐

A
```
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0
```

B
```
1 0 1 0
0 1 0 1
1 0 1 0
0 1 0 1
```

C
```
0 0 1 1
1 1 0 0
0 0 1 1
1 1 0 0
```

D
```
1 1 1 1
0 0 0 0
1 1 1 1
0 0 0 0
```

E
```
1 1 0 0
0 0 1 1
1 1 0 0
0 0 1 1
```

F
```
0 0 0 0
1 1 1 1
0 0 0 0
1 1 1 1
```

G
```
 0 1 0 1
 1 0 1
 0 1
 1
```

H
```
  1
 0 1
 1 0 1
0 1 0 1
```

I
```
1 0 1 0
1 0 1
1 0
1
```

J
```
    0
  0 1
0 1 0
 0 1 0 1
```

K
```
0 1 0 1
0 1 0
0 1
0
```

L
```
    1
  1 0
1 0 1
 1 0 1 0
```

Question 5	**Arrays**	**12** points

1. For each code fragment, write what is printed in the box on the right. If the code results in an error (compile- or run-time), write **ERROR** in all CAPS.

a.
```
int[] a = { 3, 2, 0, 1, 4 };
double[] d = {1.0, -3.14, 126.0, 0.0, 1.0};
StdOut.println(a.length == d.length);
```

b.
```
int[] a = { 3, 2, 0, 1, 4};
a = new int[3];
StdOut.println(a[1]);
```

c.
```
int[] a = { 3, 2, 0, 1, 4 };
StdOut.println(a[a[a[3]]]);
```

d.
```
int[] a = { 3, 2, 0, 1, 4 };
int[] b = { -3, -2, 0, -1, -4};
int[] c = a + b;
StdOut.println(c[1]);
```

2. Recall from Programming Exam 1, an EV charging station data file has the total number stations on the first line(≥ 1). Each subsequent line has the **`name`** and (**`x, y`**) location for a single EV station. For example:

```
% cat data3.txt
3
LocationA   40.28       -74.81
LocationB   40.38       -74.65
LocationC   40.36       -74.62
```

For each of the following problems, which solutions **require** storing all the data (using arrays) and which can be computed without using arrays? Assume all input is read from standard input using **`StdIn`**. Fill in the corresponding circle.

	Requires arrays	Does **not** require arrays
a. Plots the locations of all the stations on a map using `StdDraw`.	◯	◯
b. Prints the distances between all stations.	◯	◯

Question 6	**Input/Output/Redirection/Piping/Command-line**	**8** points

Consider these programs:

```
public class Q {
  public static void main(String[] args) {
    int num = Integer.parseInt(args[0]);
    while (!StdIn.isEmpty()) {
      int value = StdIn.readInt();
      if (value > num)
        StdOut.print(value + " ");
    }
    StdOut.println();
  }
}
```

```
public class R {
  public static void main(String[] args) {
    int a = Integer.parseInt(args[0]);
    while (!StdIn.isEmpty()) {
      int b = StdIn.readInt();
      StdOut.print(a + b);
      StdOut.print(" ");
    }
    StdOut.println();
  }
}
```

Assume the contents of the file **`input.txt`** as shown below:

```
5     1     8     9     4     6
```

Suppose that you execute each of the following commands. For each command, what numbers are printed on standard output? Write your answer in the space provided. If the command results in an error, write **ERROR** (in all CAPS). If no numbers are printed, write **NONE** (in all CAPS).

```
% java-introcs Q 4 < input.txt
```

```
% java-introcs R 1 < input.txt | java-introcs Q 5
```

```
% java-introcs Q 1 3 < input.txt
```

```
%  java-introcs R 2 | java-introcs Q 10
```

Question 7	**Functions**	**16** points

Consider the following functions:

```
public static int[] swap1(int[] a, int[] b) {
    int[] temp = b;
    a = b;
    b = temp;
    return a;
}

public static int[] swap2(int[] a, int[] b) {
    for (int i = 0; i < a.length; i++) {
      int t = a[i];
      a[i] = b[i];
      b[i] = t;
    }
    return a;
}

public static void swap3(int[] a, int[] b) {
    for (int i = 0; i < a.length; i++) {
      int t = a[i];
      a[i] = b[i];
      b[i] = t;
    }
    return;
}

public static void swap4(int[] a, int[] b) {
    b = swap1(a, b);
    swap3(a, b);
    b = new int[b.length];
    return;
}
```

Given the arrays: `int[] x = { 10, 30, 50 };` `int[] y = { 20, 40, 60 };`

Assume each function is called with the array values **x** and **y** as initialized above, where each function call is independent of one another. Fill in the circle that indicates if the array values are swapped after calling each function. Select "Not Sure" to receive 1 point partial credit. (Each part is worth 4 points.)

	Function	Does swap	Does not swap	Not sure
1	`swap1(x, y);`	◯	◯	◯
2	`swap2(x, y);`	◯	◯	◯
3	`swap3(x, y);`	◯	◯	◯
4	`swap4(x, y);`	◯	◯	◯

Question 8	**Recursion**	**10** points

Consider the following recursive static method:

```
public static String mystery(int n) {
   if (n == 0)
      return "";
   else if (n == 1)
      return "I";
   else if (n == 2)
      return "W";
   else
      return mystery(n - 1) + mystery(n - 2);
}
```

1. What is the value of **`mystery(2)`**

2. What is the value of **`mystery(3)`**

3. What is the value of **`mystery(4)`**

4. What is the value of **`mystery(6)`**

5. Provide an example of a call to **`mystery(__)`** that will produce a **`StackOverflowError`**. If there is no such example, fill in the circle for NONE.

mystery(**)** ◯ NONE

Question 9	**Performance**	**10** points

Fill in the circle that describes the order of growth of the running time with respect to **n**:

1
```
public static long mystery1(int n) {
  if (n == 1)
    return 1;
  return n * mystery1(n-1);
}
```

○ Constant ○ Logarithmic ○ Linear ○ Linearithmic ○ Quadratic ○ Cubic ○ Exponential

---

2
```
public static String mystery2(int n) {
  String s = "";
  for (int i = 0; i < n; i++)
    if (Math.random() < .5 ) s += "0";
    else                     s += "1";
  return s;
}
```

○ Constant ○ Logarithmic ○ Linear ○ Linearithmic ○ Quadratic ○ Cubic ○ Exponential

---

3
```
public static void mystery3(int n) {
  if (n == 0) return;
  StdOut.print(n + " ");
  mystery3(n-1);
  mystery3(n-1);
  mystery3(n-1);
}
```

○ Constant ○ Logarithmic ○ Linear ○ Linearithmic ○ Quadratic ○ Cubic ○ Exponential

---

4
```
public static void mystery4(int n) {
  double[] a = new double[n];
  for (int i = 0; i < n; i++)
    a[i] = Math.sin(2 * Math.PI *
                    i * 440.0 / 44100);
  StdAudio.play(a);
}
```

○ Constant ○ Logarithmic ○ Linear ○ Linearithmic ○ Quadratic ○ Cubic ○ Exponential

---

5 A program has the following running times for different sizes of its input n.

n	Time (seconds)
10000	5
20000	42
30000	135
40000	318

○ Constant ○ Logarithmic ○ Linear ○ Linearithmic ○ Quadratic ○ Cubic ○ Exponential