# CIS 1100 — Fall 2023 — Exam 1

Full Name: _____

Recitation #: _____

PennID (e.g. 12345678): _____

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 are below. Not complying will lead to a 0% score on the exam.

- Do not open this exam until told by the proctor.
- You will have exactly 60 minutes to take this exam.
- Make sure your phone is turned OFF (not on vibrate!) before the exam starts.
- Food and gum are not permitted—don't be noisy or messy.
- You may not use your phone or open your bag for any 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 in proper Java format, including all curly braces and semicolons.
- Do not separate the exam pages. Do not take any exam pages with you. The entire exam packet must be turned in as is.
- Only answers on the FRONT of pages will be graded. There are two blank pages at the end of the exam if you need extra space for any graded answers.
- 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 you.
- When you turn in your exam, you may be required to show your PennCard. If you forgot to bring your ID, talk to an exam proctor immediately.
- We wish you the best of luck!

| Q1 | Q2 | Q3 | Q4 | Q5 | Q6 (bonus) |
|---|---|---|---|---|---|
| | | | | | |

## Q1. Types Fill In The Blank

In the column marked **“Type,”** choose the type (e.g. `int`, `char`, etc.) for the variable that would allow the line to compile, or write **"compilation error"** if there is an error in the expression that makes its type undefined. You do not need to write the value of the expression.

| **Statement** | **Type** |
|---|---|
| `_____ x = "yes" + Integer.parseInt("45");` | |
| `_____ x = 5 > 8 > 6;` | |
| `_____ x = (int) "apple".charAt(0) - 'a';` | |
| `_____ x = true && 3 < 4 && !"yes".equals("no");` | |
| `_____ x = new In();` | |
| `_____ x = (int) Math.random() * 5 - 4.0;` | |
| `_____ x = new char[43];` | |
| `_____ x = 4 / 3 % 17 == "yes".equals("no")` | |

## Q2. Values Fill In the Blank

Write the value that gets printed, or write **"runtime error"** if there is an error during the execution of these lines of the program.

### Question 2.1

```
System.out.println(Integer.parseInt("3") / Double.parseDouble("4.0"));
```

Answer:

### Question 2.2

```
int[] x = {1, 2, 3};
x[x.length - 1] = x[x.length - x[0]] - x[1];
System.out.println(x[2]);
```

Answer:

### Question 2.3

```
int[] arr = {1, 2, 3};
System.out.println(arr[3]);
```

Answer:

### Question 2.4

```
System.out.println("10" + 20);
```

Answer:

### Question 2.5

```
double[] numbers = {4, 2, -1};
double x = 1;
for (int i = 0; i < numbers.length; i++) {
    x = x * numbers[i];
}
System.out.println(x);
```

Answer:

## Q3. Tracing

Here's a class that features a few functions.

```
public class TracingExercise {
    public static void main(String[] args) {
        System.out.println("Starting main");
        int x = 10;
        int y = 5;
        int z = funcOne(y, x);
        System.out.println("The final result is: " + z);
    }

    public static int funcOne(int a, int b) {
        System.out.println("funcOne arguments: a=" + a + ", b=" + b);
        int result1 = a + b;
        int result2 = funcTwo(result1);
        int result3 = funcThree(a, b);
        int finalResult = result2 - result3;
        System.out.println("funcOne returning: " + finalResult);
        return finalResult;
    }

    public static int funcTwo(int num) {
        System.out.println("funcTwo argument: num=" + num);
        int result = num * 2;
        System.out.println("funcTwo returning: " + result);
        return result;
    }

    public static int funcThree(int x, int y) {
        System.out.println("funcThree arguments: x=" + x + ", y=" + y);
        int result1 = x * 3;
        int result2 = y * 2;
        int finalResult = result1 + result2;
        System.out.println("funcThree returning: " + finalResult);
        return finalResult;
    }
}
```

When the program is run as `java TracingExercise`, eight lines are printed. For each of the following lines, fill in the blanks in the **“Printed Line”** column (on the next page) to show what values the variables have when they are printed out. Also, mark the order in which they are printed in the **“Order”** column starting at 1. The order for the first line is marked for you.

| **Printed Line** | **Order** |
|---|---|
| `Starting main` | 1 |
| `The final result is: ______` | |
| `funcOne arguments: a=_____, b=______` | |
| `funcOne returning: ________` | |
| `funcTwo argument: num=________` | |
| `funcTwo returning: ________` | |
| `funcThree argument: x=______, y=______` | |
| `funcThree returning: ________` | |

## Q4. Complete the Program: *Clock.java*

The following program is supposed to print out a time readout every fifteen minutes over the course of one day, starting with 12:00 AM, then 12:15 AM, then 12:30 AM and so on. Times in the second half of the day should be printed using the 12-hour clock, meaning that the full set of printed times should include 96 timestamps and should match the format & pattern shown in the following example:

```
12:00 AM
12:15 AM
12:30 AM
12:45 AM
1:00 AM
...       // ellipses are not printed literally;
11:30 AM  // they are here only for abbreviation.
11:45 AM
12:00 PM
12:15 PM
...
11:45 AM
```

Help write this clock program by filling in the blanks in the program on the next page.

```
public class Clock {
    public static void main(String[] args) {
        for (int hour = 0; hour < 24; ___0___) {
            for (int minute = 0; ___1___; minute += 15) {
                if (___2___ || hour == 12) {
                    System.out.print("12");
                } else {
                    System.out.print(hour % ___3___);
                }
                if (minute == 0) {
                    System.out.print(":00");
                } else {
                    System.out.print(":" + minute);
                }
                if (___4___) {
                    System.out.println(" AM");
                } else {
                    System.out.println(" PM");
                }
            }

        }
    }
}
```

| Blank # | Code |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |

## Q5. Coding: Password Hygiene

A password can be represented by a `String`. A password is said to be **strong** if it contains an uppercase letter (a `char` between 'A' and 'Z'), a lowercase letter (a `char` between 'a' and 'z'), and a digit (a char between '0' and '9'). A password is said to be **valid** if it does not contain any forbidden characters: ' ' (a space) or '-' (a hyphen). Finally, a password is **good** if it is both strong and valid. Write the following three functions: `isStrong`, `indexOfInvalidCharacter`, and `isGood`. Note the function headers & signatures for each that describe how they should behave: your functions need to return values of the correct types to receive credit! Your implementation of `isGood` must be completed in one line to receive credit, although you can still get credit for `isGood` even if your other functions are not correctly implemented.

### Question 5.1

```
/*
 * Input: a String representing the password to test
 * Output: true if the password is considered "strong"
 * and false otherwise.
 *
 * A password is "strong" if it contains an uppercase
 * letter, and a lowercase letter, and a digit character.
 */
public static boolean isStrong(String password) {









}
```

Question 5.2

```
/*
 * Input: a String representing the password to test
 * Output: the index of the position of the *first* invalid character,
 * or -1 if the String contains no invalid characters.
 *
 * A password is "valid" if it does not contain
 * either of these two characters: ' ' or '-'.
 */
public static int indexOfInvalidCharacter(String password) {









}
```

Question 5.3

```
/*
 * Input: a String representing the password to test
 * Output: true if the password is both strong and valid, false otherwise.
 *
 * A password is good if it is both strong and valid. Use the previous two
functions you wrote to complete this one in **one line only!**
 */
public static boolean isGood(String password) {




}
```

## Q6. (Bonus)

Create a piece of art (e.g. drawing, poem, short program, anything you like)!
If you are unsure of what to make, select one member of the course staff and make art about that person. All exams will get full credit for this question, no pressure to make anything in particular!

**Extra Answers Page (This page is intentionally blank)**

You may use this page for additional space for answers; keep it attached to this exam. Clearly note on the original question page that your answer is on this extra page, and clearly note on this page what question you are answering.