CIS 1100 — Fall 2022 — 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 110 minutes to take this exam.
- Make sure your phone is turned OFF (not on vibrate!) before the exam starts.
- Food and gum are strictly forbidden. Masks are optional.
- 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. You may use the back of pages for additional scratch work.
- 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 Penn Card. 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
24 pts	21 pts	18 pts	17 pts	30 pts

Q1. True & False / MCQ / Short Fill in the Blank (12 Questions)

All your answers must be placed in the answer box below each question. Only answers placed in the designated area will be graded.

Question 1.1:

True or false? The following code compiles:

```
String x = "" + 10;
```

Answer:
☐ True ☐ False

Question 1.2:

What is the value of this expression in Java? *Hint: you shouldn't need a calculator!*

```
5.439 * (10/11) * 0.93 * 2
```

Answer:

Question 1.3:

Read the following function and choose the most appropriate name based on what the function does.

```
public static String __________(String input) {
    String output = "";
        for (int i = 0; i < input.length(); i++) {
            char current = input.charAt(i);
            if (current != ' ') {
                output += current;
            }
        }
        return output;
    }
}
```

Answer:
A. countSpaces
B. duplicateSpaces
C. removeSpaces
D. reverseString

Question 1.4

What does the following code print?

```
boolean x = true;
boolean y = false;
boolean combo = (x && (y || x)) || y;
System.out.println(combo);
```

Answer:
☐ True ☐ False

Question 1.5:

Which of the following lines of code would raise a compilation error? Select all that apply.

```
int[] xs = {1, 2, 3, 4};
double[] ys = {3.4, 4.0, 3.4};
```

Answer:
A. System.out.println(xs[ys[0]]);
B. System.out.println(ys[xs[3]]);
C. System.out.println(ys[xs[xs[0]]]);
D. System.out.println(ys[xs.length - 3]);

Question 1.6:

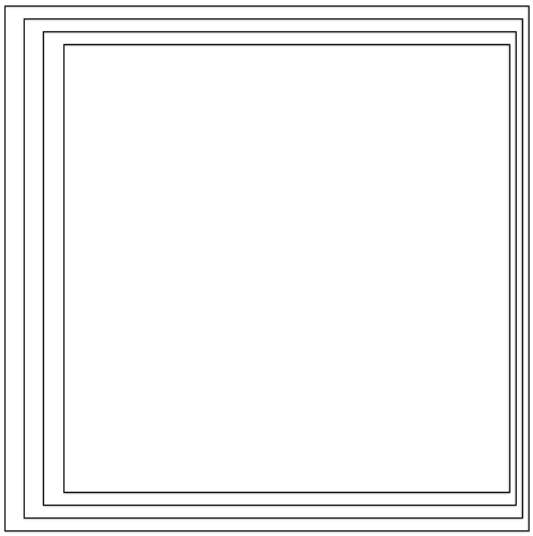
True or false? The output of this test will be true.

```
@Test
public void Test() {
       double x = 100.5;
       double y = 101.3;
       assertEquals(x, y, 1.0);
}
```

Answer:
☐ True ☐ False

Question 1.7:

Study this image and the following code template:



```
public class Rec {
    public static void main(String[] args) {
        rec(4, 0.5, 0.4);
    }

    public static void rec(int levels, double x, double r) {
        if (levels == 0) {
            return;
        }
        PennDraw.square(x, 0.5, r);
        ___RECURSIVE_CALL_________;
    }
}
```

What recursive call would generate this drawing?

Answer:

A. rec(levels - 1, x + 0.01, r - 0.02)
B. rec(levels + 1, x + 0.01, r - 0.02)
C. rec(levels - 1, x + 0.01, r)
D. rec(levels + 1, x, r - 0.02)

Question 1.8:

True or false? All functions must have a return statement.

Answer:
☐ True ☐ False

Question 1.9:

Study the following function:

```
public static _________ sumAndMultiply(int x, int y) {
    int sum = x + y;
    int result = (sum * x) + (sum * y);
    return result;
}
```

Applying your knowledge of *automatic type promotion,* decide which of the following is true about the function sumAndMultiply?

Answer:
A. The only valid return type of the function is int
B. The only valid return type of the function is boolean
C. The only valid return type of the function is char
D. char and int are both valid return types of the function
E. None of the above

Question 1.10:

I've written a function with the following signature:
`public static double distanceBetween(double[] a, double[] b)`
Which of the following functions could I write in the same class?

Answer:
A. `public static int distanceBetween(double[] a, double[] b)`
B. `public static double distanceBetween(double[] me, double[] you)`
C. `public static double similarity(double[] a, double[] b)`
D. `public static double distanceBetween(double[] b, double[] a)`

Question 1.11

Which one of the following is true about JUnit testing?

Answer:

A. JUnit is built into Java
B. It is possible that a test case passes even if the underlying code being tested contains bugs.
C. assertEquals only works for ints and doubles
D. Writing @Test before a test is not necessary, it will still be run as a test

Question 1.12:

What will the following code snippet print out?

```
System.out.println(Double.parseDouble(11));
```

Answer:

A. 11
B. 11.0
C. Does not run successfully

Q2. Long Fill in the Blank

Route Analysis

You've been tasked with helping to plan shipping routes for a fleet of trucks. You will be given data on the height in feet of underpasses on a given route. It's important to determine the lowest height of any underpass on a given route so that you can decide which model of truck to use on the route.

Given a 2D array of doubles, each row of the array represents all of the underpasses on a potential route. Each entry in the row corresponds to the height of the underpass in the route. For example, the following double[][] shows four possible routes with three underpasses each. The height of the first underpass in the first route is 9.5, and the height of the third underpass in the fourth route is 30.2. The goal is to print the underpass with the lowest height in each route (row). Fill in the blanks of the function `printAllLowest` (on next page) in the table below. For an example of how the `printAllLowest` function works, look at the following.

```
double[][] test = {{9.5, 4.8, 9.8},

                   {10.0, 9.0, 12.0},

                   {10.0, 20.0, 8.0},

                   {10.0, 13.1, 30.2}};

printAllLowest(test);
```

This is what is printed by the function:

Route 0 has lowest underpass at index 1
Route 1 has lowest underpass at index 1
Route 2 has lowest underpass at index 2
Route 3 has lowest underpass at index 0

```
public static __0__ printAllLowest(double[][] heights) {
    for (int row = 0; row < __1__; row++) {
        int indexOfMin = 0;
        __2__ currentMinHeight = heights[row][0];
        for (__3__; col < heights[row].length; col++) {
            double thisHeight = heights[row][col];
            if (thisHeight < __4__) {
                indexOfMin = col;
                currentMinHeight = thisHeight;
            }
        }
        System.out.println("Route " + __5__ + " has lowest " +
                                "underpass at index " + __6__);
    }
}
```

Blank Number	Your Answer
0	
1	
2	
3	
4	
5	
6	

Q3. Buggy Code

WilCaf has decided to implement a queue system (OHQ is being replaced by WCQ) to give customers an estimated wait time. However, a bug in the system has caused the display to calculate the incorrect wait time, thus turning away potential customers. WilCaf needs your help to solve this problem to keep business going!

The below function takes in an **integer array** where each integer value in the array indicates the amount of items a single customer ordered as a positive number. The total number of items waiting to be made is the sum of all the values in the array. Once it counts how many items are waiting to be made, the function **returns a String** that displays the wait time to show to WilCaf customers. There are 6 bugs in this code. For each bug, identify the bug and provide a brief explanation of the bug, and rewrite the line to fix the bug. (If you cannot identify every single bug, you will still get partial credit for identifying some!)

```
1    public static String wilCafWaitTime (int numItemsPerOrder) {
2         double itemCount = 0;
3         String waitTime = " ";
4         for (int i = 0; i < numItemsPerOrder.length(); i++) {
5             itemCount + numItemsPerOrder[i];
6         }
7         if (itemCount >= 0 || itemCount <= 5) {
8              waitTime = "Wait time is around 5 minutes :)";
9         } else if (itemCount >= 6 && itemCount <= 10); {
10             waitTime = "Wait time is around 10 minutes :O";
11        } else {
12             waitTime = Wait time is between 15-30 minutes :/;
13        }
14        System.out.println(waitTime);
15   }
```

Line number	Bug & Brief Explanation

Q4. Short Coding

Bhrajit and Wenny are looking to buy $X worth of boba. They arrive at Teado and are presented with an array of prices for boba.

Bhrajit and Wenny want to spend all their money and are curious about the number of ways in which they can spend their money evenly by purchasing one or zero of each menu item such that they have exactly $0 remaining. You can assume that a menu will have no items with duplicate prices, and Bhrajit and Wenny cannot buy more than one of any item.

Example: `int[] bobaPrices = {6, 7, 9, 3, 4, 10};`
Bhrajit and Wenny have 20 dollars, so there are … 4 different ways to buy boba drinks.

- 20 - 6 - 4 - 10 = 0
- 20 - 7 - 3 - 10 = 0
- 20 - 7- 9 - 4 = 0
- 20 - 6 - 3 - 7 - 4 = 0

Write a recursive function that counts the total number of ways Bhrajit and Wenny spend their money while buying distinct drinks.

Hint: you should have two recursive calls: one for including `bobaPrices[index]` and one for excluding `bobaPrices[index]`.
Hint: there is exactly **one** way to spend $0: not buying anything!

```
public static int countWays(int[] bobaPrices, int index, int money) {
    // base case(s)









    int including = countWays(_____________________________________);

    int excluding = countWays(_____________________________________);

    return including + excluding;
}
```

Q5. Long Coding: C🍪🍪KIE CRAZIE

Question 5.1: Check for Duplicates Ingredients

Insomnia cookies just had a change in management and it's absolute madness. The new manager wants to reorganize the ingredient and recipe data, and needs your help.

First complete this function to determine if there are duplicates in an array of cookie ingredients.

It is filled with Strings for types of ingredients, and the goal is to see if an ingredientList is valid, meaning it has no duplicates. The cookieList {"flour", "sugar", "butter", "chocolateChips"} has no duplicates so the function would return false. However, the list {"sugar", "butter", "sugar", "salt"} has duplicates so the function should return true on this input.

You are allowed to assume that every ingredient is given in camelCase, so you do not have to worry about checking capitalization.

```
/* Description: This function checks if there are
 * duplicates in an array
 * Input: String array of all ingredients
 * Output: Boolean of whether there exists a duplicate or not
 */
public static boolean hasDuplicates(String[] ingredientList) {
```

```
}
```

Question 5.2: Encode the Cookie Recipe

Now, we have recipes for all of the cookies. However, the new Insomnia Cookies managers are much more worried than the last about people stealing their cookie recipe. Given a single ingredient as a String, **encode** the ingredient by swapping the first and last letter and returning the result. (*Genius – now no one can figure out the recipe!)*

Example: `encode("sugar")` will return "`rugas`"

```
/* Description: This function swaps the first and last letter of a String
 * Input: String of an ingredient
 * Output: String of ingredient with first and last letter swapped
 */
public static String encode(String ingredient) {









}
```

Question 5.3: Get Ingredients' Quantities from the Encoded Recipe

Now, we have to fill out the shopping list so the Insomnia shoppers know what to buy. You are given 2 arrays, one that includes the ingredients encoded for every recipe, and one for all of the ingredients in the desired shopping list order.

The ingredientsEncoded array is a combination of every recipe. This includes the encoded ingredients for every cookie, and this is allowed to have repeats. The first thing your function should do is decode every ingredient so it is legible.

The ingredientsList array tells you the order for the quantities array.

You should only return a filled array of quantities if the ingredientsList contains no duplicates. If there are duplicates, the function should return null.

You can assume that every String in the ingredientsEncoded array, once decoded, exists in the ingredientsList array.

Example
Input:

```
String[ ] ingredientsEncoded = {“aanillav”, “rutteb”, “rlouf”,
    “rutteb”, “ehocolatc”, “rutteb”, “aanillav”};
String[ ] ingredientsList array = {“sugar”, “flour”, “butter”,
    “vanilla”, “chocolate”};
```

Output:

```
[0, 1, 3, 2, 1]
```

After you have decoded the ingredientsEncoded array, you need to count the number of each ingredient and make the count the corresponding index of the quantities array. You should return a quantities array that correctly reflects the quantity of each ingredient.

First, being the test-driven developer that you are, you want to write tests to make sure your functions work before you implement them! Please complete the following tests. There is one for each functionality, and each needs 1 assert statement.

```
@Test
public void testGetQuantities() {
    // Test case 1
    String[] ingredients = {"aanillv", "ehocolatc", "rlouf",
```

```
        "rugas", "rugas", "ehocolatc"};
    String[] list = {"vanilla", "flour," "sugar", "chocolate"};
    int[] quants = getQuantities(ingredients, list);

    // define additional variables if necessary and write case 1
    // assert statement here




}
```

```
@Test
public void testForDuplicates() {
    // test case 2
    String[] ingredients2 = {"ehocolatc", "ehocolatc", "rugas",
        "aanillv"};
    String[] list2 = {"sugar", "sugar", "vanilla", "chocolate"};

    // define any additional variables if necessary and write case
    // assert statement here




}
```

Next, implement the function!

Hint: Use code you have already written for the previous parts of this problem!

```
/* Description: This function finds the total quantity of ingredients in the
 * ingredientsEncoded array
 * Input: String array of all encoded ingredients, String array of all
 * possible ingredients
 * Output: integer array of quantities per ingredient in order of the list
of
 * ingredients
 */
public static int[] getQuantities(String[] ingredientsEncoded, String[]
ingredientsList) {









}
```

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.

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.