CIS 110 Exam 1, Summer 2021

Q1. Academic Integrity

By copying my name below, I certify that I will complete this exam in compliance with the Academic Integrity policies of CIS 110 and the University of Pennsylvania. In particular, my answers will reflect only my own work and during the exam I will only access acceptable materials including my own notes, my previous work in this class, Piazza, or the course website.

Q2. Reviewing the Building Blocks of Code

1. True or False? Any `char` value can be stored in an `int` variable, but not all `int` values can be stored in a `char` variable.

2. Fill in the most appropriate word. If code would result in an error, write "ERROR". `_____ x = 2.0 / (1 / 2);`

3. True or False? A chain of `if`/`else if` statements must finish with an `else` statement.

4.

```
public static double calculator(double a, double b, String operation) {
    double result = 0;
    switch(operation) {
        case "plus":
            result = a + b;
        case "minus":
            result = a - b;
        case "times":
            result = a * b;
        case "div":
            result = a / b;
    }
    return result;
}
```

If the line of code `double x = calculator(8.0, 2.0, "minus");` is executed, what will `x`'s value be, and why will it be this value?

5. True or False? Any code written using a `for` loop can be rewritten to use a `while` loop.

6. Fill in the blanks below so the loop does not run infinitely

```
for (int i = 10; i > 0; ____) {
    System.out.println("Hello World");
}
```

7. True or False? The line of code `PennDraw.enableAnimation(60);` makes the program wait for 60 seconds before it is allowed to continue executing code when it invokes `PennDraw.advance();`

8. In one line of code draw the image shown below, assuming no other PennDraw functions have been called beforehand:

9. True or False? As long as functions are declared inside the same class, they can access variables declared inside one another.

10. The following snippet represents a function with most of its signature missing. Fill in the blank to specify the function's return type, name, and input argument(s) so that the program will compile. If a variable is used but not declared in the function body, you can assume that it is an input argument to the function. Make sure to use good style and give the function a name that is related to its behavior.

```
public static _______ _________(________ ________) {
    double total = 0;
    for (int i = 0; i < arr.length; i++) {
        total += arr[i];
    }
    return total / arr.length;
}
```

Q3. The Fastest Cats

With the weather getting warmer and warmer in Philly, the TAs have decided to organize a race between all their cats at Clark Park. Before the race, each cat was assigned a number, and that information was stored in the String array `names`; additionally, the race time of each of the cats was stored in the double array `results`. For example, if Adam's cat "Bella" was assigned the number 3, `names[3]` would be `"Bella"` and `results[3]` would be their corresponding race time. Note that the assigned number does not necessarily correspond to their final position in the race. Since the CIS 110 TAs are very competitive when it comes to their cats, they want to know who finished first, second, and third. Help the TAs finish writing the function `findFastest`, which should return an array `fastest` with the names of fastest three cats in the order in which they finished. Note that the first index of fastest should store the name of the fastest cat. Assume there are no ties between the cats.

```
public static String[] findFastest(double[] results,
                                   String[] names) {
       double first = Double.MAX_VALUE;
       double second = Double.MAX_VALUE;
```

```
        double third = Double.MAX_VALUE;

        _____A_____[] fastest = new _____B_____[_____C_____];

        for (__________D__________) {
            if (results[i] < first) {
                third = _____E_____;
                _____F_____ = first;
                _____G_____ = _____H_____;
            } else if (results[i] < second) {
                third = _____I_____;
                second = results[i];
            } else if (results[i] < third) {
                _____J_____ = results[i];
            }
        }

        for (int i = 0; i < results.length; i++) {
            if (_____K_____) {
                fastest[_____L_____] = names[i];
            } else if (results[i] == second) {
                fastest[1] = names[i];
            } else if (results[i] == third) {
                fastest[2] = names[i];
            }
        }

        return _____M_____;
    }
```

Q4. The Great Cat Race

Yash missed the Great CIS 110 Staff Cat Race at Clark Park and wants to catch up. The Head TAs, Shanna and Shivin, posted the results on the CIS 110 website that showed him the cats (in order of finish times) and their respective times taken to complete the race. However, he wanted to see the interval times - i.e, times between the different cats.

Please help Yash write a method `public static int[] intervalTimeGenerator(int[] finalTimes)` that takes in the final race times as an int array `finalTimes` and outputs an int array of interval times.

Note: The cat who comes first will have an interval time of 0. For ex: Given an array of final times [15, 17, 25, 45], the output array of interval times will be [0, 2, 8, 20], which is calculated by [0, 17-15, 25-17, 45-25]

Note: You can assume that there are at least 2 cats in a race, but there is no upper limit.

Q5. Neko Atsume

Enzo has decided to program his own version of the mobile game Neko Atsume: Kitty Collector. As part of the game's art, he wants the cats that players can collect to have a random number of stripes in their fur. As a secret bonus to the game, if a player owns cats with even numbers of stripes and the sum of these stripes is

exactly 100, the player earns bonus in-game currency. Enzo has written a small program to determine whether a player's cat collection meets these criteria.

Unfortunately, Enzo wrote his code late at night and made a few mistakes. Identify **seven** errors in the following program that prevent it from compiling, running, or working correctly. Give the line number at which the error exists, why it is an issue, and the **correct** code with which to replace the incorrect code.

```
1  public class NekoAtsumeBonus() {
2   public static void main(String args) {
3       int numCats = args[0];
4       int[] catCollection = new int[numCats];
5       for(int i = 0; i < catCollection.length; ++i) {
6           catCollection[i] = Math.random() * 100;
7       }
8       int catEvenStripeSum = sumOfEvenStripes(catCollection);
9   }
10
11  public static int sumOfEvenStripes(int[] arr) {
12      int len = arr.length;
13      int sum = 0;
14      for(int i = 0; i <= len; ++i) {
15          if(i % 2 == 1) {
16              break;
17          }
18          sum = sum + arr[i];
19      }
20      System.out.println(sum);
21  }
22 }
```

Q6. Mauby & Dudu in the Living Room

Shanna is taking care of her friend's cat, Dudu, along with her own cat, Mauby. Dudu and Mauby are still getting to know each other, so they spend most of their time apart. Both cats love the living room, but they cannot both be in the room at the same time. To make sure both cats are happy, Shanna creates rules for living-room-time and at the end of each day, she creates a living room log to track each cat's happiness. A cat's happiness is counted as the total hours spent in the living room that day. Help her write a function to decide which cat had a happier day! She has broken down the task into smaller helper functions to make things easier. You are given an array ("living room log") `log` of Strings that contains:

- "M" if Mauby was in the living room for the hour starting at that index For example, if arr[8] = "M", then Mauby was in the living room at 8:00 Note that 0 corresponds to 12am-1am, and 23 corresponds to 11pm-12am
- "D" if Dudu was there for that hour
- "E" if the living room was empty for that hour (It is not possible for the cats to ever be together) **You can assume that this array `log` is always of length 24.**

Write the functions described below.

Shanna doesn't want the cats to have too much fun. So, the first living room rule is that no cat can spend 3 or more hours consecutively in the living room and not more than 10 hours in total in the living room in a day. You will fill in the following helper function which returns true if the `cat` spent too long at once in the living room or too long throughout the day, and false otherwise.

```
public static boolean didCatSpendTooLong(String[] log, String cat) {
    // your code here
}
```

Shanna loves sunrises and sunsets and wants to give both the cats an equal chance at spending that time with her. In Philly, the sun rises at 5:30am and the sun sets at 8:30pm. So, Shanna wants to make sure that the same cat doesn't get to spend both those hours in the living room. It is okay for the living room to be empty at that time. So, this is the second living room rule! Write a function that returns true if the same cat has been in the room at sunrise and sunset, and false otherwise. **This function can contain only one line of code.**

```
public static boolean didSameCatDoSunriseAndSunset(String[] log)  {
    // your code here
}
```

The third living room rule is that there can't be any alternate cat hours. For example, if one cat is in the room at hour 1, and the other cat uses the room for hour 2, the first cat cannot rejoin the room in hour 3. Specifically, the patterns "DMD" and "MDM" are invalid. Write the following helper function!

```
public static boolean didAnyCatAlternate(String[] log) {
    // your code here
}
```

Now that you have the 3 living room rules in place, it's time to put everything together! If the `log` array is invalid, return “None”, because she cannot fairly determine the happiness. Return the full name of the happier cat if the array is valid. Return “Tie” if the cats are equally happy and the array is valid. Note: The log is considered valid if both cats adhered to the living room rules. Here are some test cases for further understanding.

- Case 1: `{"M", "M", "D", "E", "E", "M", "D", "D", "M","M", "D", "D", "E", "M", "D", "D", "M", "E", "E", "D", "D", "M", "M", "D"}`; Output: “Dudu” This is a valid array and Dudu has spent more hours in the room than Mauby.
- Case 2: `{"M", "M", "D", "D", "M", "D", "D", "D", "M","M", "D", "D", "E", "M", "D", "D", "M", "E", "E", "E", "D", "M", "M", "D"}`; Output: “None” This is invalid because it contains the pattern DMD.
- Case 3: `{"D", "E", "D", "D", "E", "D", "D", "E", "D","E", "D", "D", "M", "E", "D", "D", "M", "E", "E", "M", "M", "D", "E", "E"}`; Output: “None” This is invalid because Dudu has spent too much time in the living room (11 hours)

- Case 4: {"M", "E", "D", "D", "E", "M", "M", "M", "E","D", "D", "E", "M", "E", "D", "D", "M", "E", "E", "M", "M", "D", "E", "M"}; Output: “None” This is invalid because the same cat was in the room for the first and last hours, and Mauby spent a consecutive 3 hours in the room.

```
public static String findTheHappierCat(String[] log) {
    // your code here
}
```