CIS 110 Summer 2021 Exam 2

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.

eg. Shivin Uppal

Q2. Reviewing the Building Blocks of Code 2.0

1. True or False? A class can have multiple constructors.

2. Fill in the most appropriate word.

```
public class Person {
    ____A____ String name;

    ____B____ String getName() {
        return this.name;
    }
}
```

3. True or False? An `interface` is only allowed to declare functions, and **not** variables.

4. Of the five variable initializations written in `main`, which one will cause a compiler error? Explain your answer in one sentence.

```
public interface Drawable {
    public void draw();
}

public class Ball implements Drawable {
    public void draw() {
         PennDraw.filledCircle(0, 0, 0.5);
    }
}

public class Box implements Drawable {
    public void draw() {
         PennDraw.filledSquare(0, 0, 0.5);
    }

    public static void main(String[] args) {
        Ball a = new Ball();
        Box b = new Box();
```

```
        Drawable c = new Ball();
        Drawable d = new Box();
        Ball e = new Box();
    }
}
```

5. True or False? The nodes of a linked list are directly adjacent to each other in memory, like how the elements of an array are.

6. Fill in the blanks below in order to iterate over all the nodes in a linked list.

```
for (Node curr = _____; _____; ____) {
    System.out.println(curr.data);
}
```

7. True or False? If one were to use a base-5 number system (in which valid digit values are 0, 1, 2, 3, and 4), the number 50 in base-10 would be represented as 200 in base-5.

8. Fill in the blanks below to complete the function, which takes a binary number represented as a `String` of 0s and 1s, and returns its base-10 numeric equivalent.

```
public static int binaryToDecimal(String binary) {
    int sum = 0;
    int powerOfTwo = 0;
    for(int i = binary.length() - 1; i >= 0; --i) {
        int digit = Integer.parseInt("" + binary.charAt(i));
        sum += _________________;
        ++powerOfTwo;
    }
    return sum;
}
```

9. True or False? A 2D array's rows (i.e. inner arrays) must all be the same length.

10. After the following code has run, what are the contents of `arr`? Your answer should be formatted similarly to how `arr` is initialized.

```
int[][] arr = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
for(int x = 0; x < arr.length; ++x) {
    for(int y = 0; y < arr[x].length; ++y) {
        if(x + y % 2 == 0) {
            arr[x][y] = 9;
        }
        else {
            arr[x][y] = arr[x][y] * 2;
        }
```

```
    }
}
```

11. True or False? Any code implemented with a `for` loop can be implemented recursively using the same amount of memory.

12. The following `public` method is intended to **return the sum of** all the powers of two up to the input power. For example, if the number `3` were input, the function would return `15`, i,.e. `2^3 + 2^2 + 2^1 + 2^0`. Fill in the blank to complete the function:

```
public int sumOfPowersOfTwo(int p) {
    if(p == 0) {
        return 1;
    }
    return _____________________;
}
```

Q3. Homework -> Homework -> Homework

The 110 staff has kept track of all of the homework assignments in the summer session by using a LinkedList represented with the following Node class:

```
public class Node {
    public Homework h;
    public Node next;
}
```

This uses the following homework class, which keeps track of both the name of the homework and the average grade that the students got on the homework:

```
public class Homework {
    public String name;
    public int avgScore;
}
```

The TAs have written a method that allows them to measure the relative difficulty of a homework assignment. The method takes in the name of a homework assignment as a `String` and returns a `Homework` in the linked list that satisfies **BOTH** of the following two conditions:

1. It has an average score **higher** than that of the homework passed into the method.

2. It has the largest difference in average score with the homework passed into the method.

In the case of a tie, return the first homework to satisfy both conditions.

Unfortunately, a sudden summer lightning storm damaged the computer the TAs' method was stored on, and now the lines of code are all scrambled, and all curly braces have been removed! Using the scrambled lines of code below, write out the method that finds and returns the `Homework` that satisfies the conditions listed above, given the `String` name of the homework assignment to compare the others to.

Note that you may not add anything of your own & must simply rearrange the lines already given to you. That means no line can be occur more than the number of times they do below, all lines must be include, and no extra lines maybe added.

You may assume that the LinkedList is not null and not empty and at least one `Homework` satisfying the conditions described exists.

```
for (Node curr = head; curr != null; curr = curr.next) {
for (Node curr = head; curr != null; curr = curr.next) {
givenGrade = curr.homework.avgGrade;
highestDifference = currDifference;
highestDifferenceHW = curr.homework;
Homework highestDifferenceHW = null;
if (curr.homework.name.equals(hwName)) {
if (curr.homework.avgGrade > givenGrade) {
if (highestDifferenceHW == null || currDifference > highestDifference) {
int currDifference = Math.abs(givenGrade - curr.homework.avgGrade);
int givenGrade = 0;
int highestDifference = 0;
return highestDifferenceHW;
}
}
}
}
}
```

```
public static Homework higherAverages(Node head, String hwName) {
    // Fill in your rearranged code
}
```

Q4. Next Stop: Broad Street!

With COVID restrictions beginning to be lifted throughout Philadelphia, SEPTA (the public transportation service of the city) needs your help with brushing off the dust from their `Service` interface, and the `BroadStreetLine` class that implements this interface. Help the SEPTA engineers find the bugs in their `BroadStreetLine` class before reopening their lines to the mass public. Identify **six** errors in the `BroadStreetLine` class 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 to write at that line. If you would like to add any code, please specify the line number **after** which you want to add your code.

For example, if this was a line of code:

```
15. int x = Math.random() * 10;
```

then you would specify that on line 15, we are trying to assign a `double` to an `int` without casting it. You would then say that the correct code is:

```
int x = (int)(Math.random() * 10);
```

Below is the code for the `Service` interface and `BroadStreetLine` class.

```
public interface Service {
    // returns the name of the service
    public String getServiceName();

    // returns max number of people allowed on vehicle
    public int getMaxCapacity();

    //returns number of people currently on vehicle
    public int getCurrentOccupancy();

    //returns array of names of stops on that line
    public String[] getListOfStops();

 }
```

```
1.  public class BroadStreetLine {
2.      private String serviceName;
3.      private int maxCapacity;
4.      private double currentOccupancy;
5.      private String[] listOfStops;

6.      public BroadStreetLine(String serviceName,
                               int maxCap,
                               int currentOccupancy,
                               String listOfStops) {
7.          serviceName = serviceName;
8.          maxCapacity = maxCap;
9.          this.currentCapacity = currentCapacity;
10.         this.listOfStops = listOfStops;
11.     }

12.     public String getServiceName() {
13.         return this.serviceName;
14.     }

15.     private int getCurrentOccupancy() {
16.         return this.currentOccupancy;
17.     }
```

```
18.     public String[] getListOfStops() {
19.         return this.listOfStops;
20.     }
21. }
```

Q5. AAAAADDDDDAAMM

Yash has an old MacBook laptop with a buggy keyboard. Every time he presses a key, the keyboard sometimes registers multiple iterations of the same character. For example, if Yash types `YASH`, the keyboard would sometimes register `YAAAAAASHHHH` or `YYYYYAAAAASSSSSHHH`. Help Yash write a recursive program to remove adjacent duplicate characters from a string, i.e, remove all consecutive characters except one. Continuing with the above example, with the input string `AAAAAADDDDDAAAAAAAMMMM` or `ADDDDDDAMMMMM`, the program will output ADAM.

Hint 1: Consider writing a private helper function to handle any additional variables you may need to track the state of during recursion.

Hint 2: Make use of the `String.substring(int index)` method, which when invoked by a `String` instance returns a new `String` containing all characters in the original **except** those prior to the given index. For example, `"ABCD".substring(2)` would return `"CD"`.

```
public static String removeDuplicates(String input) {
    // your code here
}
```

Q6. Vatsal, NYC, and Skyscrapers

Vatsal is in New York City for his internship and is fascinated by all the skyscrapers in the city. He's befriended the city cartographer who has given him access to a very interesting map. The city is represented as a 2D array where each index stores the height of the tallest building in a specific block of New York City. He has found a way to represent the city as a perfect square.

If you don't already know, there are 5 boroughs in New York City - Manhattan, the Bronx, Staten Island, Brooklyn, and Queens. Since Brooklyn has barely any skyscrapers, it is not represented in this map. So, the map is divided into the boroughs (rectangular, but not necessarily square) and Vatsal knows the 2D array indices corresponding to the corners of each borough. Each position in the 2D array contains the number of floors in the tallest building in that block.

For example, if the Queens borough has the following field variables: `xMin = 1, xMax = 3, yMin=1, yMax = 2` then it is represented by the bolded values in the following 2D array/map, and its tallest building has 102 floors.

{

{ 30, 10, 100, 40, 60, 12, 13, 14 },

{ 35, **21, 40, 45**, 72, 19, 30, 5 },

{ 17, **102, 40, 70**, 6, 12, 1, 32 },

{ 27, 14, 41, 7, 19, 55, 7, 17 }

}

So far, Vatsal has written a `Borough` class with public fields and a constructor, and a `NewYorkCity` class.

```
public class Borough {
    public int xMin;
    public int xMax;
    public int yMin;
    public int yMax;
    public String name;

    public Borough(int xMin, int xMax, int yMin, int yMax, String name) {
        // You can assume that
        // xMin, xMax, yMin, and yMax
        // are all in-bounds and valid inputs
        this.xMin = xMin;
        this.xMax = xMax;
        this.yMin = yMin;
        this.yMax = yMax;
        this.name = name;
    }
}

public class NewYorkCity {
    public int[][] map;
    public Borough[] boroughs;

    public NewYorkCity(int[][] map, Borough[] boroughs) {
    this.map = map;
    this.boroughs = boroughs;
    }

    private int heightOfTallestBuildingInBorough(Borough borough) {
        // TODO: Fill this in
    }

    public String boroughWithTallestBuilding() {
        // TODO: Fill this in
    }

}
```

Your task is to write 2 functions in total.

First, Vatsal wants you to write a function to find the number of floors in the tallest building in a specific borough. **You can assume that the `Borough borough` is not null and contains only valid data fields.** Using our example above, calling this function with our map and city should return 102.

Second, write a function that finds the `Borough` with the tallest building. Using our example above, calling this function should return `"Queens"`. Note: Vatsal wants to account for the existence of multiple dimensions of existence, so any `NewYorkCity` instance has its own distinct array of `Borough`s. As such, your program should be able to handle any arbitrary `NewYorkCity`. **Additionally, you can assume that none of the boroughs will overlap.** Because of this, you can also assume that each block is in exactly one borough.

Height of Tallest Building in Borough

Write your implementation of `heightOfTallestBuildingInBorough` below.

```
private int heightOfTallestBuildingInBorough(Borough borough) {
    // TODO: Fill this in
}
```

Borough With Tallest Building

Write your implementation of `boroughWithTallestBuilding` below. Remember to use the helper function you just wrote!

```
public String boroughWithTallestBuilding() {
    // TODO: Fill this in
}
```