CIS 110 — Introduction to Computer Programming
Fall 2019 — Exam 1

Name: _____

Recitation #: _____

Pennkey (e.g., eeaton): _____

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:

- 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, gum, and drink are strictly forbidden.
- 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, with one exception**: you <u>must</u> tear off the last two pages, as they are scratch paper. You must turn in both your exam and the separated scratch paper.** Do not take any sheets of paper with you.
- 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 ID. If you forgot to bring your ID, talk to an exam proctor immediately.
- We wish you the best of luck.

1.) The Easy One (2 pts)

Write your name on every exam page. This is in case the pages get separated during grading.

2.) General (7 pts)

For each statement below, note whether it is `true` or `false` in the space provided. The first one has been done for you as an example. Write "True" or "False". No credit will be given for "T" or "F" only.

2.0) `main()` is always the first method executed when a program is run	True
2.1) An if statement must have an else if or an else block	False
2.2) `a[a.length]` will return the last element in the array `a`	False
2.3) `(int) 13.4 + 6.6 == 20`	False
2.4) `("koala".charAt(0) - "camel".charAt(2)) == -2`	True
2.5) Variable names cannot start with numbers	True

2.6) You can have both of the following methods in one class:
`public static void launch(double power) { ... }`
`public static double launch(double accuracy){ ... }`
	False

2.7) `int[] arr = int[18];` is a valid statement	False

3.) Operators and Expressions (16 pts) – 1 point per blank

For each code fragment, (a) fill in the most appropriate data type of `z` in the 1st column and (b) give the value that `z` contains after the code has been executed in the 2nd column. If the code would result in an error, write "`ERROR`" in the 1st column and give the reason for the error in the 2nd column (you do not need to write the exact error message, just a general explanation).

The first two problems have been completed for you as examples.

Expression	**Type / ERROR**	**Value / Explanation**
`z = "4" * 4`	ERROR	* can't be used w/ `int` string
`z = 1 + 63.0`	`double`	`64.0`
`z = (int) 63.0 + 1`	int	64
`z = 3/2/1`	int	1
`z = 55 % 110`	int	55
`z = Math.sqrt(110 * 110)`	double	110.0
`z = !(!true) != !false && false || true`	boolean	true
`z = 1 + 1 + "1" + 1`	String	"211"
`z = 110 - "10" - 1`	ERROR	Bad num operation
`z = Math.random() < 1`	boolean	True

4.) Welcome to O.W.L.s (15 pts)

Hogwarts has (finally) found a way to use technology inside its premises. You, a newly recruited wizard from the muggle world, have been asked to use your coding skills to help the headmaster set up a system to assign grades to the students for their O.W.L (Ordinary Wizarding Level) exams. The grade schema is as follows:

Passing grades:

- O: Outstanding (85 - 100]
- E: Exceeds Expectations (70 - 85]
- A: Acceptable (60 - 70]

Failing grades:

- P: Poor (50 - 60]
- D: Dreadful (33 - 50]
- T: Troll [0 - 33]

"()" means exclusive and "[ ]" means inclusive (so, (70 - 85] means if someone gets a score of 85, they get an E, but if someone gets a 70, they get an A (because 70 is not included in E).

Write a function, *gradeOWLs()*, taking in a `double` array of numeric scores and returning a `char` array of their corresponding letter grades (in the same order).

Solution:

```
public static char[] gradeOWLs(double[] marks) {
    char[] grades = new char[marks.length];
    for (int i = 0; i < marks.length; i++) {
        if (marks[i] > 85) {
            grades[i] = 'O';
        } else if (marks[i] > 70) {
            grades[i] = 'E';
        } else if (marks[i] > 60) {
            grades[i] = 'A';
        } else if (marks[i] > 50) {
            grades[i] = 'P';
        } else if (marks[i] > 33) {
            grades[i] = 'D';
        } else {
            grades[i] = 'T';
        }
    }
    return grades;
}
```

5.) Game Night (15 pts)

Professor Eaton and Professor Mally are playing a game. The rules are as follows:

- The game has 9 rounds, and in each round, a random integer between 1 (inclusive) and 200 (exclusive) is generated.
    - If this integer is even, Professor Eaton gets a point.
    - If this integer is odd, Professor Mally gets a point.
    - If this integer is evenly divisible by nine, no one gets a point.
    - If the integer 110 appears, the game ends immediately.

Whoever has the most points at the end of the game wins; ties will go to Professor Eaton (alphabetically). The function below simulates this game and returns a `String` containing the name of the winning professor. Please fill in the blanks to complete the function.

```
public static String professorGame() {
    int eatonPoints = 0;
    int mallyPoints = 0;

    for (_____; _____; _____) {

        int rand = (_____) (Math.random() * _____) +
_____;

        if (rand == 110) {

            _____;

        } else if (rand % 9 == 0) {

            _____;

        } else if (_____) {
            eatonPoints++;
        } else {
            mallyPoints++;
        }
    }
    if (eatonPoints ______ mallyPoints) {
        return "Eaton";
    } else {
        return "Mally";
    }
```

```
}
```

Solution:

```
public static String professorGame() {
    int eatonPoints = 0;
    int mallyPoints = 0;

    for (int i = 0; i < 9; i++) {
        int rand = (int) (Math.random() * 199) + 1;
        if (rand == 110) {
            break;
        }  else if (rand % 9 == 0) {
            continue;
        } else if (rand % 2 == 0) {
            eatonPoints++;
        } else {
            mallyPoints++;
        }
    }
    if (eatonPoints >= mallyPoints) {
        return "Eaton";
    } else {
        return "Mally";
    }
}
```

6.) Java Jive (18 pts)

Your friends finally opened their new late night coffee shop called *Night on the Towne*! They took CIS 110 and know that a good debugging strategy is to use print statements. Tell them what prints out when they run their program to process orders from the file `orders.txt`, which has exactly the following contents:

```
medium coldBrew 2
small pumpkinSpookLatte 4
medium snowCone 1
secret candyCaneCappuccino 0
```

Assume that they ran the program by calling `java NightOnTheTowne orders.txt`

```
public class NightOnTheTowne {
  public static void main(String[] args) {
    processOrders(args[0]);
  }

  public static int totalPumps(int numFlavors, String size) {
    if (size.equals("small")) {
      return numFlavors;
    } else if (size.equals("medium")) {
      return numFlavors * 2;
    } else if (size.equals("large")) {
      return numFlavors * 3;
    } else if (size.equals("secret")) {
      return 110;
    } else {
      return -1;
    }
  }

  public static boolean isHolidayDrink(String drinkName,
                                       String[] holidayDrinks) {
    System.out.println("Checking " + drinkName);
    if (drinkName.contains("snow"))
      return true;
    for (int i = 0; i < holidayDrinks.length; i++) {
      if (drinkName.equals(holidayDrinks[i]))
        return true;
    }
    return false;
  }
```

```
  public static void processOrders(String filename) {
    String[] holidayDrinks = {"pumpkinSpookLatte",
"candyCaneCappuccino"};
    int numSongs = 5, currentSong = 2;
    boolean isHolidaySeason = false;

    In inStream = new In(filename);

    while (!inStream.isEmpty()) {
      String size = inStream.readString();
      String drinkName = inStream.readString();
      int numFlavors = inStream.readInt();

      currentSong = (currentSong * 2) % numSongs;
      System.out.println("On song " + currentSong + " of " +
numSongs);

      boolean isHolidayDrink = isHolidayDrink(drinkName,
                                              holidayDrinks);
      if (isHolidayDrink) {
        if (isHolidaySeason) {
          System.out.println("Happy holidays!");
          System.out.println("Here's your " + size + " " + drinkName
                             + " with " + totalPumps(numFlavors,
size)
                             + " pumps of syrup!");
        } else {
          System.out.println("We're out; have a free coffee.");
        }
      } else {
        System.out.println("Here's your " + size + " " + drinkName
                           + " with " + totalPumps(numFlavors, size)
                           + " pumps of syrup!");
      }
      isHolidaySeason = !isHolidaySeason;

      if (currentSong % 4 == 0) {
        System.out.println("Your drink is free!");
        numSongs++;
      } else {
        System.out.println("That'll be $" +
(numFlavors+currentSong));
      }
```

```
      System.out.println();
    }
  }
}
```

Output of `java NightOnTheTowne orders.txt`

```
On song 4 out of 5
Checking coldBrew
Here's your medium coldBrew with 4 pumps of syrup!
Your drink is free!

On song 2 out of 6
Checking pumpkinSpookLatte
Happy holidays!
Here's your small pumpkinSpookLatte with 4 pumps of syrup!
That'll be $6

On song 4 out of 6
Checking snowCone
We're out; have a free coffee.
Your drink is free!

On song 1 out of 7
Checking candyCaneCappuccino
Happy holidays!
Here's your secret candyCaneCappuccino with 110 pumps of syrup!
That'll be $1
```

7.) The Curse of Recursion (12 pts)

Michael, Max, Sierra, and Rachel all wrote recursive functions. However, because they forgot to properly comment their code, Professors Eaton and Mally have put onto them a curse (!) that can only be lifted by the correct outputs of each function being identified. The functions `maxwell()`, `sierra()`, `rachel()`, and `michael()` have each been given the arguments (`0.1`, `0.1`, `6`).

To lift the curse, match the correct TA to each of the four following outputs by writing their names below the appropriate output image.

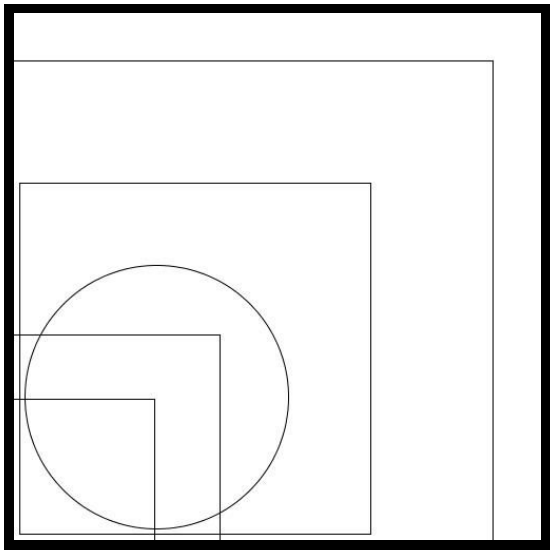

Image 1 - Maxwell

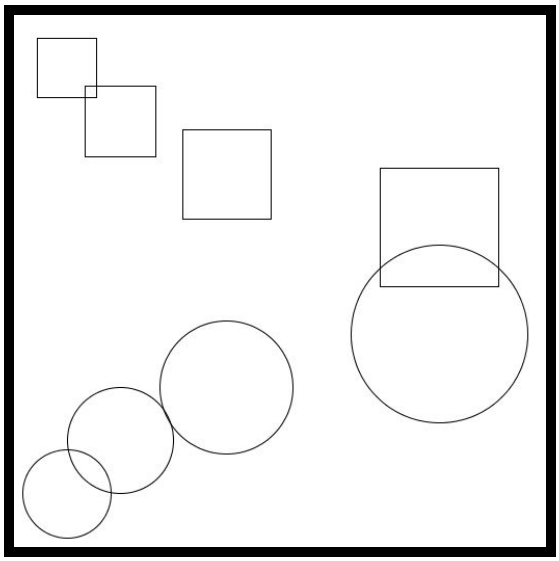

Image 2 - Michael

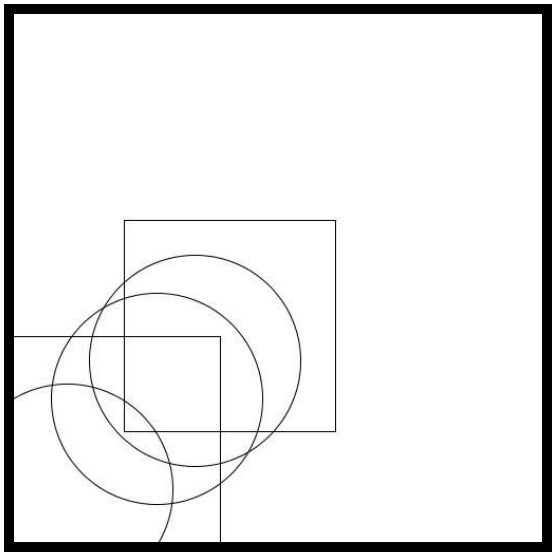

Image 3 - Sierra

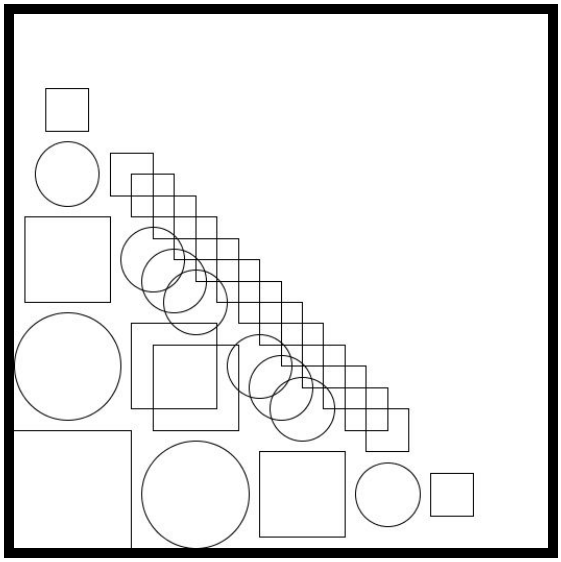

Image 4 - Rachel

```
public static void maxwell(double x, double y, int n) {
    if (n > 1) {
        if (Math.random() < 0.5) {
            PennDraw.circle(x, y, 1.0 / n);
        } else {
            PennDraw.square(x, y, 1.0 / n);
        }
        maxwell(x + (1 - x) * .1, y + (1 - y) * .1, n - 1);
    }
}

public static void sierra(double x, double y, int n) {
    if (n > 1) {
        if (Math.random() < 0.5) {
            PennDraw.circle(x, y, 0.2);
        } else {
            PennDraw.square(x, y, 0.2);
        }
        sierra(x + (1 - x) * .1, y + (1 - y) * .1, n - 1);
    }
}

public static void michael(double x, double y, int n) {
    if (n > 1) {
        PennDraw.circle(x, y, 1.0 / (2 * n));
        PennDraw.square(x, 1 - y, 1.0 / (3 * n));
        michael(x * 2, y + 0.1, n - 1);
    }
}

public static void rachel(double x, double y, int n) {
    if (n > 1) {
        double r = n * 0.015;
        if (n % 2 == 1) {
            PennDraw.circle(x, y, r);
        }
        else {
            PennDraw.square(x, y, r);
        }
        rachel(x + r * 2, y, n - 1);
        rachel(x, y + r * 2, n - 1);
    }
}
```

8.) Hellewoan Prenkstars (25 pts)

The TAs want to play a Halloween prank on the professors. Their plan is to reverse the order of vowels in every word on the lecture slides (e.g. `compiler` → `cempilor`, `method` → `mothed`). However, this would be tedious to do by hand, so they're asking you to write functions to help them. For this question, we will assume all inputs are lowercase and define vowels as only `'a'`, `'e'`, `'i'`, `'o'`, and `'u'`.

We will break this problem down into smaller tasks and write a function for each one:

1. A function to check if a character is a vowel
2. A function to count the vowels in a String
3. A function to return an array of the vowels in reverse order
4. A function to assemble the altered String

The first function we will write determines whether or not a given character is a vowel. Please complete the following function: (For full credit, your function should be only one line of code.)

```
public static boolean isVowel(char c) {




}
```

Now, using your `isVowel()` function, write a function to count the number of vowels in a `String`. (We must do this before we create an array to store the vowels, so we know how large to make the array.)

```
public static int numVowels(String s) {




}
```

Next, write a function that takes in a `String` and returns a `char` array containing the String's vowels in reverse order. You can do this by iterating over the String from the end until the beginning, and filling the array from the beginning until the end.
Hint: you should use separate index variables for the String and the array. One of these will be a for loop index variable. Also make sure to use the functions you have already written.

```
public static char[] vowelArray(String s) {




}
```

Now all we have to do is assemble the altered String from the vowel array and the original String. Once again, you will be iterating over both an array and a String, but this time you will be going through both from the beginning until the end. Complete the following `reverseVowels()` function, and be sure to use the functions you have already written:

```
public static String reverseVowels(String s) {

}
```

Solution:

```
public static boolean isVowel(char c) {
    return c == 'a' || c == 'e' || c == 'i' ||
        c == 'o' || c == 'u';
}

public static int numVowels(String s) {
    int count = 0;
    for (int i = 0; i < s.length(); i++) {
        if (isVowel(s.charAt(i))) {
            count++;
        }
    }
    return count;
}

public static char[] vowelArray(String s) {
    char[] vowels = new char[numVowels(s)];
    int index = 0;
    for (int i = s.length() - 1; i >= 0; i--) {
        if (isVowel(s.charAt(i))) {
            vowels[index] = s.charAt(i);
            index++;
        }
    }
    return vowels;
}

public static String reverseVowels(String s) {
    char[] vowels = vowelArray(s);
    String out = "";
    int index = 0;
    for (int i = 0; i < s.length(); i++) {
        if (isVowel(s.charAt(i))) {
            out += vowels[index];
            index++;
        } else {
            out += s.charAt(i);
        }
    }
    return out;
}
```

Extra Answers Page

(This page 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 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.

Scrap Paper

(This page intentionally blank)

You **must detach this page from the exam**, and you **must turn it in**.

Scrap Paper

(This page intentionally blank)

You **must detach this page from the exam**, and you **must turn it in**.