# CIS 110 — Introduction to Computer Programming
# Fall 2017 — Midterm

Name: ____________________

Recitation ROOM : ____________________

Pennkey ***(e.g., paulmcb)***: ____________________

**DO NOT WRITE YOUR ID# ABOVE, YOU WILL LOSE A POINT**

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 120 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts. You will lose 5 points if your cell phone goes off.**
- Food, gum, and drink are strictly forbidden.
- **You may not use your phone or open your bag for <u>any</u> 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 out in proper java format, including all curly braces and semicolons.
- **<u>Do not separate the pages.</u>** You may tear off the one scratch page at the end of the exam. This scratch paper must be turned in or you lose 2 points per page.
- Turn in all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper for scratch work, please use the backs of the exam pages or the extra pages provided at the end of the exam. **Only answers on the FRONT of pages will be grading. The back is for scratch work only.**
- 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 answer them.
- 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.**

**Scores:** [For instructor use only]

| | | |
|---|---|---|
| Question 1 | | 6 pts |
| Question 2 | | 16 pts |
| Question 3 | | 10 pts |
| Question 4 | | 7 pts |
| Question 5 | | 11 pts |
| Question 6 | | 4 pts |
| Question 7 | | 11 pts |
| **Total:** | | **65 pts** |

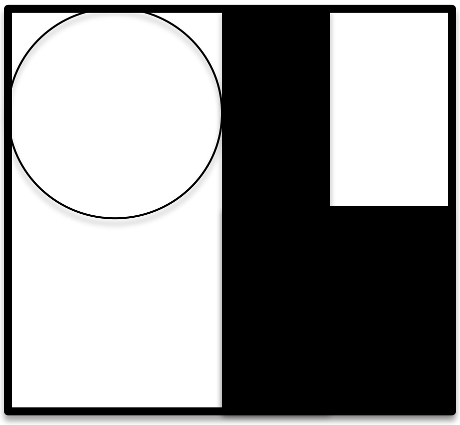
## 1.) Miscellaneous (6 points)

**1.1) (1 point)** The Easy One:

- Check that your exam has all 10 pages (excluding the cover sheet, which is not numbered).
- Write your name, recitation number, and PennKey (username) on the front of the exam.
- Sign the certification that you comply with the Penn Academic Integrity Code.

**1.2) Draw the result of the following code (do not worry about color): (3 points)**

```
PennDraw.circle(0.25, 0.75, 0.25);
PennDraw.filledSquare(0.75, 0.25, 0.25);
PennDraw.filledRectangle(0.6, 0.5,
                         0.1, 0.5);
```



**1.3) Write the following while loop as a for loop that produces the same output: (2 points)**

```
int x = 15;
while (x > 0) {
     System.out.println(x – 1);
     x = x / 2;
}
```

```
for (int x = 15; x > 0; x = x /2) {
     System.out.println(x – 1);
}
```

**2) Datatypes (16 points)**
In the table below, the left column will list some code involving a variable X. You must fill out the two columns on the right. You must identify the data type of the variable X (from data types be have mentioned in class) and the value of X after the code is executed. However, the code may have errors. If the code results in a ***compile time error***, write "CE" under "Type of X" and explain the error under "value of X". **(1 point per box) Instructions corrections: If the code results in a runtime error, write "RE" under "Type of X" and explain the error under "value of X".**

| **Code** | **Type of x** | **Value of x** |
|---|---|---|
| `___ x = 5;` `x = 7;` | **int** | 7 |
| `int a = 6.5;` `___ x = a;` | CE | Cannot assign 6.5 to an integer. |
| `String s = "abc";` `___ x = s.charAt(s.charAt(0) -` `s.charAt(2));` | **RE** | Index out of bounds, since it checks for s.charAt(-2); |
| `int y = 5 / 3;` `____ x = int y++;` | CE | y declared twice |
| `String s = "abc";` `char y = 'a';` `__ x = (s + y) - y;` | CE | Cannot subtract from a String. |
| `boolean a = false;` `boolean b = !(a && true);` `__ x = (!a && !b) \|\| (a \|\| !(b && a))` | boolean | true |
| `__ x = (int) ((int) 4.9) + 2.9` | double | 6.9 |
| `__ x = {"a", "b", "c", "d"}` `x[0]++;` | CE | Cannot increment a String with ++ |
| `int y = Double.parseDouble("3");` `char z = '3';` `__ x = y == z;` | CE | Cannot convert Double to Integer automatically |
| `int y = 5 % 2;` `int z = 9 % 3;` `__ x = y + y + z + " is fun!"` | String | "2 is fun!" |

**3) Loops and Arrays (10 points)**
With all the events going on the world today, many people are finding that they can't even. Can't even keep up with it. Can't even cope. Just **can't even**. Luckily, as computer programmers, we are here to help. The below function takes in an array of numbers from the command arguments and returns an array with only odd numbers.

Example, if the input array were [10, 8, 7, 2, 3, 4], the output of function would be [7, 3]

Unfortunately, the programmer writing this code **can't even** finish. So fill in the blanks

```
public static int[] cantEven(int[] input) {
      //find the number of odd values in the inputArray
      int numOdd = 0;

      for (int i = 0; i < input.length; i++){


            if (input[i] % 2 == 1) {
                  numOdd++;
            }
      }

      //create an array of odd values


      int[] odds = new int[numOdd];
      int counter = 0; //where we next insert into odds array



      for (int i = 0; i < input.length; i++){



            if (input[i] % 2 != 0) {
                  odds[counter] = input[i];



                  counter++; //increment counter
            }
      }



      return odds;
}
```

**4) Tracking Wild Animals (7 points)**
A bunch of wild animals have invaded this code, making it very hard to read. Trace your way through this code and below write exactly what would print to the console in the box below.

```
public class DangerousAnimals {
    public static void main(String[] args) {
        double[] zebra = new double[16];
        for (int lion = 0; lion < zebra.length; lion++) {
            if (lion % 2 == 1 && lion > 9) {
                zebra[lion] = lion * 1.5;
            }
            else if (lion < 7) {
                zebra[lion] = lion * 2;
            }
            else {
                zebra[lion] = lion + (lion % 2);
            }
        }

        for (int rhinoceros = zebra.length - 3;
               rhinoceros >= "hippo".length();
               rhinoceros -= 2) {
            System.out.println((int) zebra[rhinoceros]);
        }
    }
}
```

```
19
16
10
8
10
```

**Hint: The end result of zebra after the first loop is:**

**0, 2, 4, 6, 8, 10, 12, 8, 8, 10, 10, 16.5, 12, 19.5, 14, 22.5**

**5) Recursion can be fun! (11 points)**

Some students really do not enjoy learning recursion. So to help, we made a fun recursion function. And to illustrate this fact, we called it "fun". This function does not do any meaningful mathematical operation, so it's just for fun.

```
public static int fun(int x, int y, int z) {
        System.out.println(x + " " + y + " " + z);
        if (z != 0) {
            return fun(y, x, 0);
        } else if (y == 0) {
            return x;
        } else if (x == 0) {
            return y;
        } else {
            return fun(y/2, Math.abs(y-x), 1);
        }
}
```

Notice the print statement on line one. In the boxes below, write what prints for each function call. Hint You are given the correct number of lines. Then in the box to the right of it, write the final return value of this call.

`fun(2, 0, 0)` — **Return value**

```
1.2,0,0
```

Return value: 2

`fun(3, 3, 0)`

```
1.3,3,0

2.1,0,1

3.0,1,0
```

Return value: 1

`fun(2, 2, 1)`

```
1.2,2,1

2.2,2,0

3.1,0,1

4.0,1,0
```

Return value: 1

## 6) Bug Infestation (4 points)

An exterminator has been cataloguing the bugs she finds by putting them into arrays (before… well…exterminating them). She gives each bug a number, and if she finds a second type of the same bug, it gets the same number. For example, a termite might be a 0, an ant 1, a cockroach 2, etc.

So if her array is `[0,2,1,1,2,2,0,0]` it means she found 3 termites, 2 ants, and 3 cockroaches. After she cleans out a house, she takes her array of bugs back to her computer to find out how many of each bug there was.

She wrote a function, `numOccurrences()` below that would take the above array, and a range of bug ID numbers, and return the number of each bug.

Examples:
`numOccurrences({0,2,1,1,2,2,0,0}, 3)` should return an array `[3, 2, 3]`
`numOccurrences({2,2,0,2,3,2}, 4)` should return an array `[1, 0, 4, 1]`

Unfortunately, our stalwart exterminator must now deal with bugs of a different kind. She wrote the following code to count the number of each bug, but her code is infested with…well…bugs:

```
1    public static int[] numOccurrences(int[] arr, int range) {
2         int[] numOcc == new double[arr.length];
3         for (int i = 0; i <= arr.length; i++) {
4              numOcc[arr[j]]++;
5         }
6         return arr;
7    }
```

Rewrite the code in the box below **with the same structure and number of lines of code.** Just fix the bugs in each line (there are bugs on lines 2, 3, 4, and 6.). Do not write your own function, but instead fix the function above.

```
public static int[] numOccurrences(int[] arr, int range) {
     int[] numOcc = new int[range];
     for (int i = 0; i < arr.length; i++) {
          numOcc[arr[i]]++;
     }
     return numOcc;
}
```

**7) Building a Better Sportsball (11 points)**

You may not know this, but Dr. McBurney is the inventor of the greatest of outdoor games: Sportsball. In sports ball, teams of two play against each other, where every player's skill is denoted by an integer. The larger the integer, the better the player. However, ticket sales have been dwindling in the NSL (National Sportsball League) due to one team (the Topeka Fightin' Fighters) dominating the league and making it uninteresting to watch. To rectify this, Will has decided to turn to the world of code to make sure the league will be competitive. And he's recruiting you to salvage his league before a bunch of unpaid Sportsball players break his legs.

Specifically, you are to write a function isEqualEnough with the following declaration:

```
public static boolean isEqualEnough(int[] players, int inequalityFactor) {
```

`players:` an array of integers, with each integer denoting the skill of one sportsball players.
`inequalityFactor:` an integer denoting the maximum allowed inequality.

Returns false if the greatest sum of any two integers is greater than ineqaulityFactor times the smallest sum of any two integers. You can assume all individual strengths and inequalityFactor are positive, and that there are an even number of players and the length of players is >= 2.

Example: If players = [1, 3, 8, 1] and inequalityFactor = 3, the greatest possible sum is 11 and the smallest 2. Since 11 is greater than 2 * 3, this **returns false**.

Example: If players = [4, 3, 8, 5] and inequalityFactor = 2, the greatest possible sum is 13, and the smallest 7. Since 13 is **NOT** greater than 7 * 2, this **returns true.**

**Use the next two pages to write this function, but do NOT write any code you wish to have graded on the back of any page.**

Note that there are tons of ways to do this. This is just one way.

```
public static boolean isEqualEnough(int[] players,
                                   int inequalityFactor) {
        int min = players[0] + players[1];
        int max = players[0] + players[1];
        for (int i = 0; i < players.length; i++) {
            for (int j = 0; j < players.length; j++) {
                if (i != j) {
                    max = Math.max(max, players[i] + players[j]);
                    min = Math.min(min, players[i] + players[j]);
                }
            }
        }
        return !(max > min * inequalityFactor);
  }
```

**Extra Space for Answers**

**DO NOT RIP THIS PAGE OFF. ANY WRITING ON THIS PAGE CAN BE GRADED**