

## Assignment 5 - More Java Practice

**Problem 1.** Take a positive integer like  $n$ . If it is even, divide it by 2. If it is odd, multiply it by 3 and add 1 to it. Repeat this process until you get 1. For example, if we start with 10, the sequence goes:  $10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$ . In this problem we will write a method called `collatz` that prints this sequence to the terminal.

a) Fill in this method body to solve this problem iteratively.

```
public static void collatz(int n) {  
    // We need a variable for the number we are current only, how do we declare that?  
  
    -----  
  
    // We should first print out our current number before applying the rule  
    // How do we do that?  
  
    -----  
  
    // We need a loop, but given an int $n$ we don't know how many times we will need to iterate.  
    // What kind of loop should we use? What should it's condition be?  
  
    ----- {  
  
        // Now how do we do something if our # is even vs if it is odd? How do we check those?  
        // Remember the % operator!  
  
        ----- {  
  
            // our # is even here, how do we update the number?  
  
            -----  
  
        } ----- {  
  
            // our # is odd here, how do we update the number?  
  
            -----  
  
        }  
  
        // After getting our new number, we should print it out  
  
        -----  
  
    }  
}
```

b) Write `collatzRecursive` that solves this problem recursively. What is your base case? When you aren't in your base case, what recursive call(s) are you going to make?

**Problem 2.** The following code snippet has many issues including both syntax errors, typos, and logical errors. Identify as many as you can and explain the mistake. Rewrite the method to do as described.

```
/**
 * Takes, as input, two Strings a and b, as well as an int c
 * It prints out (to the terminal) c + 10 if a is the string "bye"
 * If not, and b doesn't start with the letter "c", it prints c^2
 * If neither are true it prints c
 */
String ERROR_METHOD{a, string b} {
    if a == bye (
        System.out.println(c + 10)
    ) if (b.charAt(1) == 'c') {
        System.in.println(c^2);
    }
    return System.out.println(this.c);
}
```

**Problem 3.** Fill in the following method that returns whether or not the given string is a *palindrome* (is the same string when reversed).

```
public static _____ isPalindrome(String s) {

    int length = s.length()

    // Loop through the string, checking if 1st character matches the last,
    // if the 2nd character matches the second to last, etc.

    // We do not need to loop through the entire string
    // When can the loop stop? Try working it using an explicit example:
    // If you check "BANANA" as described, what iteration do you stop at?

    for (_____ ; _____ ; _____) {

        // we want to compare the i-th character with it's corresponding
        // spot on the opposite half of the string. Can you work out the
        // formula for which character corresponds with character i?
        // Work it out by doing some explicit examples

        if (s.charAt(i) != s.CharAt(_____)) {
            return _____ ;
        }

    }

    // All the checks in the loop passed, what do we return?

    return _____;

}
```

**Problem 4.** Below are several array problems. The problems are not in order of difficulty. If you are unable to work out a part, move on to the next part. If you need to call a method defined in a previous subpart, pretend you defined it correctly.

a) Write a public static method `copy(int[] arr)` that returns a *new* array that is equal to `arr` (i.e. has the exact same contents), using a for loop.

**Warning:** You need to instantiate a new array and copy the values one-by-one. You cannot just return `arr`, this does not copy the array.

```
public static badCopy(int[] arr) {
    return arr;
}

int[] a = {1, 2, 3};
int[] b = badCopy(a);
b[0] = 10;
System.out.println(a[0]); // Prints 10, a is now {10, 2, 3}
```

This behavior is different than with variables that store *primitives* due to technical details of Java's implementation. Primitive variables *store their data directly* while reference type variables (like arrays) store a *reference* (memory address) to where their data is stored. So `return x` where  $x = 5$ , returns the actual *value* 5, but `return arr` in `badCopy` returns the memory address of the array that `arr` points to. So `a` and `b` in the code above both point to the same array in memory. Hence modifying `b` also modifies `a` (and vice versa).

b) Write a public static void method `swap(int[] arr, int i, int j)` that modifies `arr` by swapping the position of the  $(i + 1)$ th and  $(j + 1)$ th elements. For example,

```
int[] a = {1, 2, 3, 4, 5};

swap(a, 1, 4);

System.out.println(a[1]) // Prints 5
System.out.println(a[4]) // Prints 2, a = {1, 5, 3, 4, 2}
```

The “issue” described above allows modifications to arrays to persist outside of the methods that make the modification. This does not happen with primitive variables!

```
public void test(int a) {
    a = 10;
}

int x = 5;

test(x);

System.out.println(x) // Prints 5, x is still equal to 5.
```

c) Write the method `public static void reverse(int[] arr)` that reverses `arr`. Two ways to do this:

1. Create a `tempCopy` array that is a copy of `arr`. Then loop over `arr` and copy the contents of `tempCopy` in reverse order.
2. Use `swap` and loop through `arr`, swapping the first and last, then the second and second to last, etc. This way is similar to a previous problem on this homework.

Both are good practice!

d) Write a method `public static boolean equals(int[] arrA, int[] arrB)` that returns whether two `int` arrays store the same data in the same order using a `for` loop.

**Hint:** Your code should first check if the two arrays are the same size and return `false` immediately if not. This is called a “guard clause,” because it “guards” the bulk of your code from a specific case. It looks like this:

```
... methodName(...) {
    if (guardCondition) {
        return ...; // Or throw an exception
    }

    // Rest of your code, which is not wrapped in an else case
    // This means we don't need to indent all of our code
}
```

e) An array is called *palindromic* if it reads the same way forwards as backwards.<sup>1</sup> Use three of the methods you defined above to write a `public static isPalindromic(int[] arr)` method, that returns whether `arr` is palindromic. For example `isPalindromic({0, 1, 0})` returns `true` and `isPalindromic({1, 2})` returns `false`.

**Hint:** You can do this in one line

---

<sup>1</sup>Source: I made it up!

**Problem 5.** Consider the following class:

```
public class Point {  
    public double x; // x coordinate of the Point  
    public double y; // y coordinate of the Point  
}
```

**a)** Add a constructor to this class body that takes in two doubles xPos and yPos and constructs a Point that represents a point at (xPos, yPos)

**b)** Add a `public static` method called `distanceBetween` that takes in two Points a and b and returns the distance between the two points (as a double). Recall that for two points  $(x, y)$  and  $(a, b)$ , the distance between them is given by:

$$\sqrt{(x - a)^2 + (y - b)^2}$$

Look at the documentation for Math (<https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>) for helpful methods for computing this value

**c)** Write a public *instance* (non-static) method `distanceTo` that takes in a Point a, and returns the distance between a and *this* point.

Hint: You can use the method you just defined above with the correct inputs. You may find the `this` keyword helpful for this.

**d)** Use the method you just defined to write a public instance method `isOnUnitCircle` that returns whether or not *this* point is on the unit circle (i.e. exactly distance 1 away from the origin (0,0)).

**WARNING:** Arithmetic with doubles is not precise. `1.0/3.0 * 3.0` may evaluate to 0.99999998 or 1.000000001 (neither of which is 1.0). To check if a double *a* is equal to another *b*, we should check if their difference is very small, say  $|a - b| < 0.000001$ . Use `Math.abs(a)` to compute  $|a|$ , the absolute value of *a*

**Hint:** Remember, in order to call a non-static method from this class, we use the dot operator on the `this` keyword (e.g. `this.methodName()`).