0. BERN, JAMES

CS136 Midterm

Fall 2023

- **Do not turn this page until time is called.**
- We will read through the info below together. 🙂👍

- The exam is 100 points total.
- The exam is scheduled for 90 minutes.
- No notes are allowed.

- Just like in the homework, we are compiling with Java 8 and importing java.util.*
- Assume code snippets are inside of suitable main methods.
- We read binary numbers from right-to left.
    - 110 in binary means ((1 * 4) + (1 * 2) + (0 * 1)) = (4 + 2) = 6

- This exam may be curved, but only up (the curve cannot make your grade worse).
- This exam will be primarily graded on correctness.
- This exam may also be graded (to a lesser extent) on speed, clarity, and robustness.

```
// better
if ((c >= '0') && (c <= '9')) { ... }
// worse
if ((c == '0') || (c == '1') || (c == '2') || (c == '3')
     || (c == '4') || (c == '5') || (c == '6')
     || (c == '7') || (c == '8') || (c == '9')) { ... }
```

```
// better
int[] foo = new int[100];
foo[foo.length / 2] = 1;
// worse
int[] foo = new int[100];
foo[50] = 1;
```

Please sign below.

I am the person whose name is listed at the top of this page.
I have neither given nor received unauthorized help on this exam.

____

1. **(20 points) 1D Elementary Cellular Automata.**

HINT: Here are the numbers 0 through 7 written as **3-digit binary numbers**.

	7	6	5	4	3	2	1	0
	111	110	101	100	011	010	001	000

HINT: Here is 22 = (16 + 4 + 2) written as the 8-digit binary number 00010110.

	0	0	0	1	0	1	1	0
	128	64	32	16	8	4	2	1

HINT: Here is Rule 22, which we get by putting the previous two hints together.

current	111	110	101	100	011	010	001	000
next	0	0	0	1	0	1	1	0

a. **What is 37 as an 8-digit binary number? Answer goes in the empty row below.**

	7	6	5	4	3	2	1	0	
current	111	110	101	100	011	010	001	000	
next									<- Answer goes here.
	128	64	32	16	8	4	2	1	

b. **Using Rule 37, evolve this automata 5 generations into the future.**
   i. **The top row contains the starting state.**
   ii. **The far left and far right cells are treated special and stay dead forever.**

0	0	0	1	0	0	0
0						0
0						0
0						0
0						0
0						0

2. **(20 points) Make the code below support 4 Player’s.**
    a. **Store the 4 Player's in an array; use a for loop to avoid repeating code.**
    b. **Make each Player have the same radius and same position.**
    c. **Make each Player have a different color.**
    d. **Assume you have the same Player class as in Homework-03's starter code.**
    e. **Just modify the code below, you do NOT need to implement update(), etc.**

```
class HW03 extends App { // BEFORE
    Player player;

    void setup() {
        player = new Player();
        player.color = Vector3.cyan;
        player.radius = 4.0;
        player.position = new Vector2(0.0, 0.0)
    }
}
```

```
class HW03 extends App { // AFTER (Answer goes inside here.)

}
```

3. (20 points) Implement this function, which returns an `ArrayList<Integer>`.

HINT: An integer is *even* if it is divisible by 2.
HINT: `(x % y == 0)` is a `boolean` that says whether x is divisible by y

```
// Returns the indices (NOT values) of all even numbers in the array.
// NOTE: Assume all numbers in the array are >= 0.
//
// { 6, 7, 5, 8 } -> { 0, 3 } because the 0th element (6)
//     and 3rd element (8) are even
//
// { 7, 1, 5 } -> {} (empty list) because no elements are even
ArrayList<Integer> getIndicesOfEvenNumbers(int[] array) {

}
```

4. **(20 points) Arrays and array lists.**

   a. **What will this code print? Answers go next to the print statements.**

```java
int[] array = { 2, 3, 4 };

ArrayList<Integer> list = new ArrayList<>();
for (int i = 0; i < array.length; ++i) {
    for (int repetition = 0; repetition < 3; ++repetition) {
        list.add(array[i]);
    }
}

System.out.println(list); //                      <- Answer goes here

for (int i = 0; i < list.size() / 2; ++i) {
    int j = (list.size() - 1) - i;
    int tmp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, tmp);
}

System.out.println(list); //                      <- Answer goes here

for (int i = 0; i < list.size(); ++i) {
    list.set(i, list.get(list.get(i))); // <- Read this carefully!
}

System.out.println(list); //                     <- Answer goes here.
```

   b. **Assume that `list`'s initial capacity (length of its internal/private array) is 1, and that `list` grows (when necessary) by doubling its capacity.**
      After the code above runs, what is `list`'s capacity?
      `// Answer goes here:`

5. **(15 points) What will this code print? Answers go next to the print statements.**

```java
int i = 3;
boolean b = (i < 7);
System.out.println(b); //            <- Answer goes here.
b = !b;
System.out.println(b); //            <- Answer goes here.
char c = 'A' + ('3' - '0');
System.out.println(c); //            <- Answer goes here.
```

6. **(5 points) Finish implementing this function.**

```java
// call an array of integers coprime if 1 is the only divisor that
//     ALL numbers in the array have in common
// { 4, 6, 5 } IS coprime because 1 is the only divisor all numbers
//     have in common (NOTE: 4 and 6 are divisible by 2, but 5 is NOT)
// { 3, 6, 9, 12, 33 } is NOT coprime because all numbers in the array
//     are divisible by 3
// NOTE: Assume all numbers in the array are >= 2.
static boolean isCoprime(int[] array) {









}
```