Name: ____

USC Student ID Number:__________________________________

USC NetID (e.g., `ttrojan`):_____________________________

CS 455 Midterm Exam 1
Spring 2022 [Bono]

Tuesday, Feb. 15, 2022

There are 6 problems on the exam, with 60 points total available. There are 10 pages to the exam (5 pages **double-sided**), including this one; make sure you have all of them. If you need additional space to write any answers or scratch work, pages 9 and 10 are left mostly blank for that purpose. If you use those pages for answers, you just need to direct us to look there. ***Do not detach any pages from this exam.***

Any remote students printing the exam should scan, and submit all pages of the exam, even if you didn't write on a particular page. If you print it single-sided, do not write on the backs of pages and do not scan blank backs of pages.

Note: if you give multiple answers for a problem, we will only grade the first one. Avoid this issue by labeling and circling your final answers and crossing out any other answers you changed your mind about (though it's fine if you show your work).

Put your name, USC Student ID, and USC username (a.k.a., NetID) at the top of the exam. Also put your NetID at the top right of the front side of each page of the exam (including the last page). Please read over the whole test before beginning. Good luck!

`Arrays toString` method documentation: (for problem 1)

```
public static String toString(int[] a)
```

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets (`"[]"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space).

Problem 1 [10 pts. total]

Consider the following program: (Note: more about `Arrays.toString` on the cover page of the exam.)

```
public class Prob1 {
   public static void foo(int[] nums) {
      nums[1] = 17;
      nums = new int[6];
      nums[2] = 12;
   }
   public static void main(String[] args) {
      int[] myNums = new int[3];
      for (int i = 0; i < myNums.length; i++) {
         myNums[i] = 2*i;
      }
      foo(myNums);
      System.out.println(Arrays.toString(myNums));
   }
}
```

Part A [6]. In the space below, draw a single box-and-pointer diagram (a.k.a., memory diagram) showing all object variables, objects, and their state as they have changed while running the program. This includes showing what happens inside of `foo`.

Part B [4]. What is printed by the code?

USC NetID: ___________________________

Problem 2 [8 pts. total]

The following method doesn't work. The method comment describes what it is supposed to do.

```
/**
   Creates a String containing the elements of the array, nums, in an addition
   expression. If nums has just one element, the String just has that one number.
   If nums.length = 0, returns an empty string.
   Example Strings for various calls (shows return value for arrays with 1, 2, and
   3 elements, respectively):
       "5"
       "4 + 9"
       "8 + 10 + 6"
*/
public static String makeAdditionExpression(int[] nums) {

   String expr = "";


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


      expr = expr + nums[i] + " + ";


   }


   return expr;


}
```

Part A [2]. Show two example data sets for which the method returns the wrong result and what the method returns in that case:

nums	**printAdditionProblem(nums)**
1.	
2.	

Part B [6]. Fix the code above. Do not rewrite the whole method, but rather make your changes right into the code above, using arrows to show where your code should be inserted, crossing out code that you would get rid of, etc.

Problem 3 [12 pts]

Complete the implementation of the class `FibonacciGenerator`, which generates the Fibonacci sequence. A solution that stores the whole Fibonacci sequence so far in the object will not receive full credit.

The Fibonacci sequence starts as follows: 1, 1, 2, 3, 5, 8, 13, . . .

Let's call the k'th Fibonacci number f_k. Using this notation, the rules for generating the sequence are outlined below:

$f_1 = 1$

$f_2 = 1$

for any $k > 2$, $f_k = f_{k-1} + f_{k-2}$

So, in the sequence shown above, the third value is 1 + 1 = 2, the fourth value is 1 + 2 = 3, and the fifth value is 2 + 3 = 5, etc.

Here is some example code that uses this class. Given a value, *n*, `printFibs` uses our class to print out the first *n* Fibonacci numbers, one per line:

```
// PRECONDITION: n > 0
public static void printFibs(int n) {
   FibonacciGenerator fib = new FibonacciGenerator();
   for (int i = 1; i <= n; i++) {
      System.out.prinln(fib.next());
   }
}
```

Turn the page for space for your answer (includes the class interface for you) →

Problem 3 (cont.)

```
/**
   Generates the Fibonacci sequence.
*/
public class FibonacciGenerator {









   /**
      Create a Fibonacci Generator
   */
   public FibonacciGenerator () {









   }


   /**
     Generate the next number in the Fibonacci sequence.
   */
   public int next () {


















   }

}
```

Problem 4 [8 pts. total]

Consider a class with the following interface:

```
//   Stores a sequence of integer data values and supports some computations
//   with it.
public class Nums {
   //  Create an empty sequence of nums.
   public Nums () { . . . }

   //  Add a value to the end of the sequence.
   public void add(int value) { . . .  }

   //  Return the minimum value in the sequence.
   //  If the sequence is empty, returns Integer.MAX_VALUE
   public int minVal() { . . . }

   //  Returns a String representation of the sequence of values formatted
   //  as a space-separated list of numbers surrounded by parentheses.
   //  E.g., "(3 7 4 10 2 7)"; for empty sequence: "()"
   public String toString() { . . . }

   //  Returns the number of values in this Nums object
   public int getSize() { . . . }
}
```

Suppose a friend of yours has started to work on an implementation for this class, and chose the following representation:

```
   private ArrayList<Integer> vals;     // the values
   private int numVals;                 // number of values we're currently storing
```

Part A [3]. Write a representation invariant for the given instance variables:

Part B [2]. You point out to your friend that this choice of instance variables has a redundancy issue. Show an updated declarations of instance variables that fixes that problem (Note: the old representation invariant won't necessarily apply to the new version).

Part C [3]. Explain why your change is preferrable to the original (no credit for "less redundant").

USC NetID: ____________________________

Problem 5 [7 points]

Come up with a thorough set of test cases to test the `minVal` method of the `Nums` class from problem 4.

For each test case:

- show which `Nums` value you are testing (we'll call it `nums` here) by showing what would be returned by calling `toString` on that object (reminder of format: a space separated list of numbers in parentheses).
- provide the expected result for that call to `minVal`,
- describe what case you are testing

`nums.toString()`	**expected `nums.minVal()`**	**description of test case**

Problem 6 [15 points]

Implement the static `zeroCrossings` method, which takes an array of ints called `nums`, and returns an `ArrayList` containing the locations of all the zero crossings in `nums`. For the location of the crossing use the actual index where it crosses or, if it crosses between two adjacent values, the midpoint between the two array indices (see examples below). For this problem we're interpreting **`nums`** as a sequence of values to be plotted in a line graph, where the x-coordinate for a value is the index of the array element containing that value (the value is the y-coordinate), and a zero crossing indicates (roughly) a place where the graph would cross the x axis. **To make this problem simpler, you should assume `nums` has the following structure:**

- **it won't start or end with a 0**
- **it will never have two or more more adjacent 0's** (e.g., [3, 0 2] ok; but [3, 0, 0, 2] not ok)

Here are some examples:

`nums`	`zeroCrossing(nums)`	
`[-1, -1, -4]`	`[]`	
`[]`	`[]`	
`[2, 4, 4, 5, 3, 2]`	`[]`	
`[2, 0, 4, 5, 0, 3]`	`[]`	(touches, but does not cross)
`[-1, 0, -5]`	`[]`	
`[-1, 0, 5]`	`[1]`	
`[3, 4, -1, -2]`	`[1.5]`	
`[2, -4, -4, 0, 4, -5, 3, 0, 3, -2]`	`[0.5, 3.0, 4.5, 5.5, 8.5]`	

```
// PRE: see bold bulleted list above
public static ArrayList<Double> zeroCrossings(int[] nums) {
```

Extra space for answers or scratch work.

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.