Name: _____

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

CS 455 Midterm Exam 1
Spring 2016 [Bono]

Thursday, Feb. 18, 2016

There are 5 problems on the exam, with 59 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 blank for that purpose. If you use these pages for answers you just need to direct us to look there.

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 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. Please read over the whole test before beginning. Good luck!

Selected methods of Java `Point` class:

`new Point(x, y)`

Constructs point object with given x and y values.

`p.translate(dx, dy)`

Changes `x` and `y` values of `p` by `dx` and `dy`, respectively. I.e., if `p` had coordinates `(x, y)`, its new value is a point with coordinates `(x+dx, y+dy)` [this is a mutator]

Problem 1 [10 pts.]

Consider the following program. Note: it uses the Java `Point` class, which is mutable – more information about `Point` on the cover of the exam.

```
public class Prob1 {
  public static void main(String[] args) {
    Point p = new Point(3, 5);
    Point q = p;
    Point r = p;
    p.translate(2, 2);
    foo(r);
    q = new Point(8, 15)
    System.out.println(p + " " + q + " " + r);
  }

  public static void foo(Point s) {
    s = new Point(10, 20);
  }
}
```

Part A [7]. In the space below, draw a box-and-pointer diagram (a.k.a., memory diagram) showing all object variables, objects, and their state as they have changed during the code sequence. This includes showing `foo`'s parameters.

Part B [3]. What is printed by the code? For the purpose of this problem assume a `Point` is printed as follows: `[x, y]`

USC NetID: _____

Problem 2 [6 pts.]

Consider the following static method that is supposed to return the letter grade for a student with the given score. It doesn't always do the right thing.

```
// PRE: 0 <= score <= 100
public static char getGrade(int score) {
   if (score > 90) { return 'A'; }
   if ((score > 80) && (score < 90)) { return 'B'; }
   if ((score > 70) && (score < 80)) { return 'C'; }
   if ((score > 60) && (score < 70)) { return 'D'; }
   return 'F';
}
```

Do not modify the code. **Show two example data values and the result of calling the method on each of them: the first one should be one where the existing method returns an incorrect grade, and a second one such that the method returns a correct grade:**

score	return value of **getGrade(score)**

1. (wrong)

2. (right)

Problem 3 [8 pts. total]

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

```
/**
   Prompts for and reads a non-negative integer from the user.  If the value
   entered is negative, prints an error message and prompts again, until the user
   enters a non-negative value.
*/
public static int readVal(Scanner in) {

  int value = -1;

  while (value < 0) {

    System.out.println("ERROR: value entered was < 0.");

    System.out.print("Please enter a non-negative integer: ");

    value = in.nextInt();

  }

  return value;

}
```

Part A [2]. Show a sample interaction (i.e., input and output), and return value for this method that illustrates the problem.

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.

USC NetID: _____

Problem 4 [15 pts.]

Complete the implementation of the class `Car`, defined on the next two pages and described further below.

Consider a Car class that keeps track of gas and miles driven, assuming a particular miles per gallon fuel efficiency for the car. You specify the MPG in the constructor; the car starts out with no gas in the tank. It will keep track of the fuel level and miles driven as the car get's driven (the `drive` method), or as gas gets put in the tank (the `addGas` method), and you can ask about the fuel level and the miles driven. If you try to drive farther than the amount of gas in the tank will allow, it will only go as far as the amount of gas you have left. Below is an example of the use of this class:

```
public static void main(String[] args) {

   Car myCar = new Car(20);         // my car gets 20 miles per gallon (MPG)

   System.out.println(myCar.gasLeft());    // 0.0: gas tank starts out empty
   myCar.addGas(5.0);                      // put 5 gallons in the tank
   myCar.drive(80);                        // drive 80 miles (returns 80.0)
   System.out.println(myCar.gasLeft());    // 1.0 gallon left
   System.out.println(myCar.milesDriven());  // 80.0
   double distance = myCar.drive(50);      // attempt to drive 50 more miles
          // returns actual distance driven before running out of gas (20.0)

   System.out.println(myCar.gasLeft());      // 0.0
   System.out.println(myCar.milesDriven());  // 100.0
   distance = myCar.drive(10);           // returns 0.0 (already out of gas)
   myCar.addGas(2.0);
   myCar.drive(30);
   myCar.addGas(3.0);
   System.out.println(myCar.gasLeft());      // 3.5
   System.out.println(myCar.milesDriven());  // 130.0

   . . .
}
```

The class interface and method comments appear on the next two pages. We left space so you can add in your code to complete the class.

(problem continued on the next page)

Problem 4 (cont.)

```
//   Car keeps track of gas and miles driven, assuming a particular miles
//   per gallon fuel efficiency for the car.
public class Car {









   //   Creates a car with the given miles per gallon (MPG).  It starts
   //   out with no fuel in the tank and no miles driven.
   public Car (int milesPerGallon) {









   }


   // Returns the amount of gas in the tank, in gallons
   public double gasLeft() {









   }

   // Returns the number of miles driven since this object was created.
   public double milesDriven() {









   }
```

(problem continued on the next page)

Problem 4 (cont.)

```
   // Add the given amount of gas to the tank.
   // (NOTE: no limit on gas tank size.)
   // PRECONDITION: gallons >= 0.0
   public void addGas(double gallons) {









   }


   // Attempt to drive the distance given in miles.  If we run out of gas
   // before going that distance, returns the actual distance driven,
   // otherwise returns miles (value given in parameter).
   // PRECONDITION: miles >= 0
   public double drive(int miles) {









   }
}
```

Problem 5 [20 pts.]

Write the static Java `int` method, `longestRun`, which takes an array of numbers and returns the length of the longest run in the array. A run is a sequence of one or more occurrences of the same value all next to each other. An array with no elements has a longest run of 0.

Examples

`nums`	**`longestRun(nums)`**
[4, 12, 4, 4]	*2*
[10, 10, 10]	*3*
[20, 20, 5, 10, 10, 18]	*2*
[5, 5, 3, 3, 3, 3, 7, 7, 7]	*4*
[12, 8, 17]	*1*
[32]	*1*
[]	*0*

```
public static int longestRun(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.