Name: _____

USC Student ID Number: _____

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

CS 455 Midterm Exam 1
Spring 2023 [Bono]

Tuesday, Feb. 14, 2023

There are 7 problems on the exam, with 61 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 an answer 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!

Reminder of some Linux shell commands and other shell syntax: (problem 4)

`ls cd cp mv rm mkdir rmdir more diff ~ * .. < > . /`

Selected methods of Java `Point` class: (problem 2)

`new Point(x, y)`	Constructs point object with given `x` and `y` values.
`p1.equals(p2)`	Returns true iff `p1` and `p2` have the same `x` and `y` values.

Problem 1 [4 pts. total]

Consider the following static method that is supposed to return the letter grade for a student with the given score based on 90, 80, 70, 60 cutoffs. 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 a correct grade, and the second one such that the method returns an incorrect grade:**

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

1. (right)

2. (wrong)

USC NetID: _____

Problem 2 [9 pts. total]

Consider the following program:

```
public class BoxAndPointer {
   public static void main(String[] args) {
      Point p = new Point(10, 7);
      Point t = new Point(10, 6);
      Point r = new Point(10, 7);
      Point s = r;
      r = p;
   }
}
```

Part A [5]. 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.

Part B [4]. Assume each of the following 8 boolean expressions is evaluated at the end of `main`, above. Circle the the correct value for each.

1. `s == p` [TRUE / FALSE]
2. `s == t` [TRUE / FALSE]
3. `s == r` [TRUE / FALSE]
4. `p == r` [TRUE / FALSE]
5. `s.equals(p);` [TRUE / FALSE]
6. `s.equals(t);` [TRUE / FALSE]
7. `s.equals(r);` [TRUE / FALSE]
8. `p.equals(r);` [TRUE / FALSE]

Problem 3 [7 points]

Consider the following method, whose interface is given below:

```
// makeWords takes a String containing some text and converts it into
// an ArrayList of individual words in the text.  By "word" we mean a
// sequence of non-whitespace characters.
// PRE: text != null
public static ArrayList<String> makeWords(String text) {
```

First, write a thorough set of test cases for `makeWords` (your solution should include at least four distinct cases). For each one, give the `text` parameter value, the expected return value, and a description of the general case your input represents. We gave one test case as an example; your cases will have to be different than that one. Second: add a description for the test case we gave. Show all `Strings` in double-quotes, and show `ArrayList`'s using the format illustrated in test case 0.

`text` parameter	**expected return value**
`"Hey there, girl!"`	`["Hey", "there,", "girl!"]`

description:

Problem 4 [4 points]

Suppose you are currently in a directory called `ex2` which is within a directory called `lab3` in your home directory in a Vocareum workspace. Copy only the *source code files* from `ex2` (shown below) to a *new* directory called `ex3` that will also be in the `lab3` directory. Your complete answer must use three or fewer Linux commands. (Linux syntax hints given on front page of exam.)

The commands below (using $ as the Linux prompt) show the starting conditions:

```
$ cd
$ cd lab3
$ ls
ex2
$ cd ex2
$ ls
CashReg2.class CashReg2.java CashReg.class CashReg.java CashRegTester.class
CashRegTester.java ex2.out
```

Problem 5 [2 points]

When we compile the `BankAccount` class below we get several compile errors with the message "cannot find symbol". Fix the problem so the code compiles and works correctly. Make your changes right into the code below, using arrows to show where any new code should be inserted, crossing out code that you would get rid of, etc.

```
public class BankAccount {
   public BankAccount() {
      int balance = 0;
   }
   public BankAccount(int initBalance) {
      int balance = initBalance;
   }
   public void deposit(int amount) {
      balance += amount;
   }
   public void withdraw(int amount) {
      balance -= amount;
   }
   public int getBalance() {
      return balance;
   }
}
```

Problem 6 [15 points]

Complete the implementation of the `RoachPop` class (skeleton below and continuing to the next page), which simulates changes in a roach population. For full credit, use the named constants provided in the class. Here's some more information about how `RoachPop` works:

A `RoachPop` starts out at a certain size, which can be affected by breeding (which doubles it size), spraying (which reduces its size by 20%), and by setting traps (which attracts and kills 25 roaches per trap). Roach populations can never go below zero. Note: It's fine if percentages are not rounded. Here is an example of using it:

```
public static void main(String[] args) {
   RoachPop roaches = new RoachPop(1000);
   roaches.breed();   // 2000
   roaches.spray();   // 1600
   roaches.breed();   // 3200
   roaches.setTraps(10);  // 2950
   System.out.println(roaches.getSize());    // 2950
   System.out.println(roaches.percentOfStartPop() + "%");   // 295%
   roaches = new RoachPop(1);
   roaches.setTraps(1);
   System.out.println(roaches.getSize());    // 0
   System.out.println(roaches.percentOfStartPop() + "%"); // 0%
}
```

The class itself, and space for your answer starts here and continues onto the next page:

```
//   Simulate changes in a roach population.
public class RoachPop {
   public static final int TRAP_CAPACITY = 25;
                     // number of roaches a trap can kill
   public static final int SPRAY_PERCENT = 20;









   //  Creates a roach population of the given size
   //  PRE: size > 0
   public RoachPop(int initSize) {




   }


   //   Gets the current population size
   public int getSize() {




   }
```

(problem continued next page)

Problem 6 (continued)

```
   // The current population as a percentage of the original population.
   // (so a value < 100 means the roach population decreased)
   public int percentOfStartPop()









   }


   // The population breeds
   public void breed() {









   }


  //  The population gets sprayed
   public void spray() {









   }


   //  Sets traps for the roaches.
   //  PRE: numTraps > 0
   public void setTraps(int numTraps) {









   }

}
```

Problem 7 [20 points]

Implement the static Java `int` method `maxDistance`, which takes an array and a value and returns the maximum distance between instances of `value` in the array. Return 0 if the value doesn't appear in the array. The definition of distance of a value in an array: a single instance of a value has a distance of one from itself, and the distance between two instances is the number of elements between them, inclusive.

Here are some examples:

`nums`	`value`	`maxDistance(nums, value)`
`[2,3,6,10]`	4	0
`[9,9,9,9,9]`	9	5
`[5,2,3,2,3,2,3,2,0,4]`	2	7
`[3,2,7,2,1]`	2	3
`[1,5,5,7]`	5	2
`[1,5,5,7]`	7	1
`[]`	7	0

```
public static int maxDistance(int[] nums, int value) {
```

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.