Name: _____

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

CS 455 Midterm Exam 1
Fall 2016 [Bono]

Thursday, Sept. 29, 2016

There are 5 problems on the exam, with 56 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 those pages for answers you just need to direct us to look there. *Do not detach any pages from this exam.*

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 value after the call 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 – more information about `Point` on the cover of the exam.

```
public class Prob1 {
  public static void foo(Point c, Point d) {
    c = d;
    d.translate(5, 10);
    System.out.println(c + " " + d);
  }
  public static void main(String[] args) {
    Point a = new Point(12, 24);
    Point b = new Point(3, 7);
    foo(a, b);
    System.out.println(a + " " + b);
  }
}
```

Part A [6]. 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 [4]. 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 a `String` describing the weather, when given an outside temperature in Fahrenheit. (Approximately equivalent temperatures are also shown in Celsius for those of you who aren't used to Fahrenheit.) It doesn't always do the right thing.

```
public static String getWeather(int temp) {
  if (temp >= 90) { weather = "boiling"; }          // 90 is about 32 C
  if (temp < 90 && temp >= 80) { weather = "hot"; } // 80 is about 27 C
  if (temp < 80 && temp >= 70) { weather = "just right"; }// 70 is about 21 C
  if (temp < 70 && temp >= 60) {                    // 60 is about 16 C
     weather = "cool";
  }
  else {
     weather = "cold";
  }
  return weather;
}
```

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 weather description, and a second one such that the method returns an accurate weather description:**

	temp	return value of **getWeather(temp)**
1.	(wrong)	
2.	(right)	

Problem 3 [10 pts.]

Implement the class `DigitExtractor`, which breaks up an integer into its individual digits. After being initialized with a positive integer, each call to `nextDigit()` returns the next digit in the integer, starting from the *leftmost* (most significant) digit. For full credit do not call `largestPowerOf10` (details below) more than once in the lifetime of a `DigitExtractor` object.

Here is an example of code to use the class. It extracts all of the digits of 27,054:

```
DigitExtractor extractor = new DigitExtractor(27054);
while (extractor.hasNextDigit()) {
   System.out.println(extractor.nextDigit());
}
```

Corresponding output:

```
2
7
0
5
4
```

You may assume we have already written the following helper method for you that works as described: (i.e., that means you may call this method in your solution)

```
// Returns the largest power of 10 that is less or equal to num.
// or put another way: returns a power of 10 that has the same number of
// digits as num
// PRECONDITION: num > 0
//   Examples:   num      return value of largestPowerOf10(num)
//                999          100
//               1000         1000
//               1001         1000
private static int largestPowerOf10(int num) { . . . }
```

For full credit do not call `largestPowerOf10` more than once per `DigitExtractor` object.

`DigitExtractor` hint: use the modulus operator (`%`) and integer division:

- The modulus operator gives the remainder of dividing two integers. For example, the result of

  `17 % 3`

  is `2` (17 divided by 3 has a remainder of 2).

- Reminder: Integer division is done with the / operator performed on two integers. For example, the result of

  `17 / 3`

  is `5`.

[Do not write your answer here. The class interface and space for your answer is provided on the next page.]

Problem 3 (cont.)

Complete the implementation of `DigitExtractor` – details of this problem are on the previous page.

```
// DigitExtractor breaks up a positive integer into its individual digits.
public class DigitExtractor {

   // Creates digit extractor for the given integer.
   // @param anInt the integer to extract from
   // PRECONDITION: anInt > 0
   public DigitExtractor(int anInt) {

   }

   // Returns true iff there are more digits left to extract.
   public boolean hasNextDigit() {

   }

   // Extracts the the "next" digit in the integer (starts from leftmost
   // (most significant) digit, and goes rightward)
   // PRECONDITION: hasNextDigit()
   // @return the digit
    public int nextDigit() {

    }

    // largestPowerOf10: see comment on previous page
    private static int largestPowerOf10(int num) { /* already written */ }
}
```

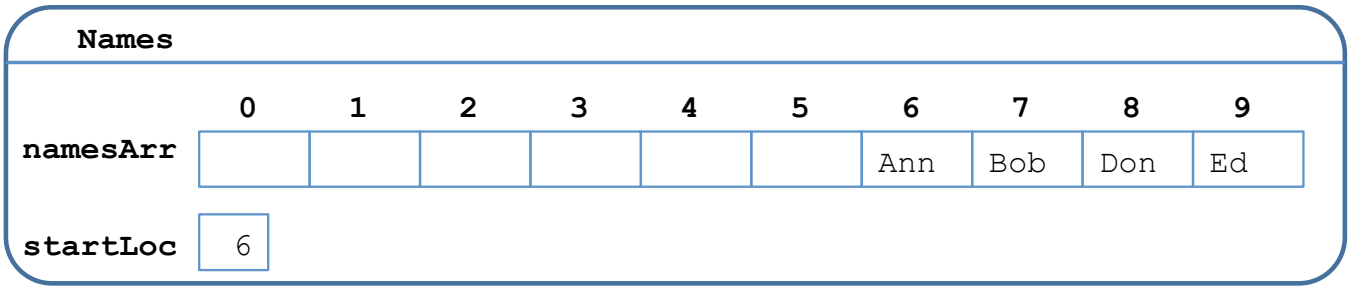
Problem 4 [25 pts. total]

Consider a slightly different representation for the `Names` class than the one we used in lecture. This new version will use a partially filled array, but the array will be filled from the *right* side instead of the left, although the values will still be in alphabetical order (not reverse alphabetical).

Here are the instance variables for this representation:

```
private String[] namesArr;
private int startLoc; // location of the first name in namesArr
```

Here is an example of such an object containing the names `Ann, Bob, Don, Ed`:



Part A [5]. Most of a representation invariant for this class appears below. Fill in the blanks to complete it so it matches the representation described and illustrated above. Hint: the representation invariant is in general terms; it should not involve any of the particular numbers from the example object above.

- the number of names is _____
- if the names object isn't empty the names are in array elements:

  `namesArr[`_____`]` through

  `namesArr[`_____`]`

- the valid range for *`startLoc`* is: _____
  (you can show as an expression with relational operators such as < or <=, or using [..,..] or [..,..), for example, range notation.)
- the names in *`namesArr`* are in alphabetical order from left to right
- the names in *`namesArr`* are unique

Part B [20]. Implement the constructor, `numNames`, and `remove` methods of the `Names` class that appear on the next page (space for your answer is there too) using the representation described here.

Note: You may use the helper method `lookupLoc` in your solution (specification shown on next page). Do not implement `lookupLoc`.

Problem 4 (cont.)

Part B (cont). Implement the following methods of the `Names` class, using the representation discussed on the previous page (you do not have to complete the rest of the class):

```
// Stores a list of unique names in alphabetical order.  Allows
// look-up, insert, and removal of elements in the list.
public class Names {

   private String[] namesArr;
   private int startLoc;   // location of the first name in namesArr

   private final static int NOT_FOUND = -1;
   private final static int INITIAL_CAPACITY = 10;
                                          // starting capacity of array
   // Creates an empty names object
   public Names() {



   }

   //  Returns the number of names in the list
   public int numNames() {




   }

   //  Removes target from names, and returns true.
   //   If target wasn't present in names, returns false and no change
   //   made to names.
   public boolean remove(String target) {

   }

   //   lookupLoc returns index of target in namesArr or NOT_FOUND
   //          if it is not present
   private int lookupLoc(String target) { . . . } // do not implement
   . . . // other Names methods not shown
}
```

Problem 5 [5 points]

The following method to print out a sentence doesn't quite work as desired: it prints a space between the last word and the period at the end of the sentence. The method comment describes what it is *supposed* to do.

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

```
/**
   Prints out an arraylist of words as a period-terminated
   sentence with a single space between each word.
   Here are some examples to illustrate the output format:
      words                    output of printSentence(words)

      ["Hello"]                   Hello.
      ["Hello", "there"]          Hello there.
      ["I", "am", "a", "cat"]     I am a cat.

   PRECONDITION: words.size() > = 1
*/
public static void printSentence(ArrayList<String> words) {


   for (int i = 0; i < words.size(); i++) {


     System.out.print(words.get(i) + " ");


   }


   System.out.println(".");


}
```

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.