Name: _____

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

CS 455 Midterm Exam 2
Fall 2015 [Bono]

Nov. 10, 2015

There are 9 problems on the exam, with 54 points total available. There are 8 pages to the exam (4 pages double-sided), including this one; make sure you have all of them. If you need additional space to write any answers, pages 7 and 8 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!

Additional Java library information:

`Map.Entry<KeyType, ValueType>` Interface

`KeyType getKey()`	Return the key of the entry
`ValueType getValue()`	Return the value of the entry
`void setValue(newVal)`	Replace the current value with `newVal`

Select Java `Scanner` and `File` constructors:

```
public File(String pathname)
```

Creates File object from given pathname. Does not open any file on disk.

```
public Scanner(File source) throws FileNotFoundException
```

Constructs a new `Scanner` that produces values scanned from the specified file. `FileNotFoundException` (a checked exception) is thrown if source file is not found on disk.

Problem 1 [2 points]

Not counting any eliminated in the first iteration, roughly how many elements are eliminated from further consideration in the **second** loop iteration of a binary search in an array of *n* elements (i.e., assuming we didn't find the value in that iteration)?

Problem 2 [2 points]

What is true about the first *k* elements in the array before pass *k* of insertion sort? (circle the answer that best describes what is known)

a. they are *k* values in sorted order

b. they are the *k* smallest values

c. they are the *k* smallest values in sorted order

d. none of the above

Problem 3 [2 points]

Suppose you are building a system to manage scientists' access to a supercomputer. What abstract data type discussed in this course would you use to keep track of scientists' computing jobs that are waiting to be processed so that they will be processed in the same order they came in?

Problem 4 [4 points]

Both the **Java** `ListIterator` interface, used for iterating over a `LinkedList`, and the `Iterator` interface, used for iterating over a `Set`, include methods `hasNext()` and `next()`.

In addition, *only* the `ListIterator` interface has the following additional method (where `ElmtType` is the type of an element in the `LinkedList`):

```
// Replaces the last element returned by next with the specified element
public void set(ElmtType newVal) . . .
```

Why is this method not provided in the `Iterator` interface?

Problem 5 [4 points]

Suppose someone proposes the following bad hash function for a hash table that has Strings as keys. What's wrong with it? [Note: the code compiles, it has the correct parameter type and return value, and returns a value in the correct range. Also the comments below match the code given.]

```
public static int hash(String key) {
   Scanner in = new Scanner(System.in);
   int alpha = in.nextInt();
// e.g., for 3-char String with chars c0,c1,c2: computes (alpha)^2*c0 + alpha*c1 + c2
   int h = 0;
   for (int i = 0; i < key.length(); i++) {
       h = alpha * h + (int) key.charAt(i);
   }
   return h % HASHSIZE;  // HASHSIZE is the size of the hash table array
}
```

Problem 6 [3 points]

The following code doesn't compile, producing an error message related to exceptions on the line marked with *****, below. **Modify the code below so it compiles.** Don't rewrite the code: make your modifications directly to the code below. (Notes: (1) There is more than one correct answer. (2) Do not worry about import statements. (3) See first page of exam for more on the classes/methods used here.)

```
public class ConcordFromFile {

   public static void main(String[] args) {

      Scanner in = new Scanner(System.in);

      System.out.print("Please enter the name of the text file for concordance: ");

      String fileName = in.next();

      File inFile = new File(fileName);

      Scanner fileScanner = new Scanner(inFile); // *

      Concord concord = new Concord();

      concord.addData(fileScanner);

      concord.print(System.out);

   }
}
```

Problem 7 [9 pts total]

Consider the following code to compute the number of occurrences of each score from an array of scores:

```
// Returns an array of the number of occurrences of each score from
//   scores arraylist.
//   E.g., after we call it as follows:
//          int[] counts = getCounts(scores, 100);
//   counts[0] is the number of 0's in scores,
//   counts[1] is  the number of 1's in scores, . . . ,
//   counts[100] is the number of 100's in scores
// PRE: all the elements in scores are in the range [0,maxScore]
public static int[] getCounts(ArrayList<Integer> scores, int maxScore) {

   int[] counts = new int[maxScore+1];  // all values init'd to 0

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

      for (int score = 0; score <= maxScore; score++) {

         if (scores.get(i) == score) {

            counts[score]++;

         }

      }

   }
   return counts;
}
```

Part A [2]. What is the worst case big-O time for the **current** `getCounts` method in terms of `scores.size()` and `maxScore`?

Part B [5]. Modify the `getCounts` function to improve the big-O running time. You should make your modifications directly in the code above, using arrows to show where your new code should be inserted, crossing out code that you would get rid of, etc.

Part C [2]. What is the worst case big-O time for your **new version** of `getCounts`?

Problem 8 [11 points]

Shown here are the specifications of the `printSatisfying` method of the `Concord` class, and the `Predicate` interface, both of which we used in a recent lab. (See front page of exam for additional information on `Map.Entry`):

```
public class Concord {  // Concord stores a collection of words and their frequencies
    private Map<String, Integer> concord;
    . . .
    // printSatisfying writes some entries to out, using same format as Concord
    // print method.  The entries it writes are the ones that satisfy pred.
    // @param out the outstream to write to
    // @param pred the predicate that each entry is tested on.
    public void printSatisfying(PrintStream out, Predicate pred) {…}

    . . .  // other details of this class left out

}

/* Describes any class whose objects can test something about a Map.Entry
 *    containing a String and Integer.
 */
interface Predicate {
    // Tests whether item satisfies some condition.
    // @param item the entry to be tested
    // @return whether this predicate is true about the entry
    boolean predicate(Map.Entry<String,Integer> item);
}
```

Using `printSatisfying` and the `Predicate` interface shown above, implement the function `printDivByK`, which prints out all the `concord` entries whose number of occurrences is divisible by `k` (they should get printed to `System.out`). Hint: Besides the method itself, your solution will need to include a class that implements the `Predicate` interface; furthermore, this class will likely need a constructor. Note: `printDivByK` is *not* part of the `Concord` class.

```
public static void printDivByK(Concord concord, int k) {
```

Problem 9 [17 points]

Write the static method `makeString`, which takes an array of words and returns a `String` containing all the words, each space terminated, using recursion. Hint: You will need to use a helper method to do the actual recursion. **Two of the points for this problem will be assigned for writing a clear and accurate method comment for your helper method.** Such a comment will also help you in writing your code correctly (to aid in thinking recursively). Your method comment does not have to appear at the start of the method as long as you label it and circle it wherever you do write it.

Examples (arrays shown below in `toString` format, as comma-separated with brackets):

`words`	***return value of call to `makeString(words)`***	
`["the","big","dog"]`	`"the big dog "`	*(note space after the last word)*
`[]` (i.e., empty array)	`""`	
`["the"]`	`"the "`	*(note space after the last word)*

Little or no credit will be given for a solution that does not use recursion.

```
// returns a String made up of the words in words array, each space terminated.
// if the words array is empty, returns an empty string.
public static String makeString(String[] words) {
```

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.