Name: _____

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

Midterm Exam 2
CS 455, Spring 2015

April 7, 2015

There are 7 problems on the exam, with 60 points total available. There are 8 pages to the exam, including this one; make sure you have all of them. There is also a separate one-page code handout. For on-campus students: if you need additional space to write any answers, you may use the backs of exam pages (just 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 NetId at the top of the exam. Please read over the whole test before beginning. Good luck!

	value	score
Problem 1	6 pts.	
Problem 2	7 pts.	
Problem 3	5 pts.	
Problem 4 & 5	10 pts.	
Problem 6	15 pts.	
Problem 7	17 pts.	
TOTAL	60 pts.	

Problem 1 [6 points]

Consider the use of a stack for storing information about method activations as a program runs. As discussed in lecture this is called the *run-time stack* or *call stack*. Reminder: a stack has the operations push, pop, top, and isEmpty.

Part A. In this application of stacks, calling a method corresponds to which stack operation?

Part B. In this application of stacks, returning from a method corresponds to which stack operation?

Part C. Consider the following program and suppose we are currently executing at the point labeled ****C****. Show *two* different possible contents of the run-time stack at this point. Notes: Make it clear which direction the stack is going by labeling which end is the top of each stack you draw.

```
public class StackProb {
  public static void main(String[] args) {
    W();
    X();
    Y();
  }
  public static void W() {
    X();
    Z();
  }
  public static void X() {
    Y();
    Z();
    // **C**
  }
  public static void Y() {
    Z();
  }
  public static void Z() {
    return;
  }
}
```

Problem 2 [7 pts.]

Consider the following static method (Note: see the code handout for a reminder of `ArrayList` methods):

```
// PRE: nums.size() >= 1
public static ArrayList<Integer> foo(ArrayList<Integer> nums) {
   ArrayList<Integer> tmp = new ArrayList<Integer>();
   tmp.add(nums.get(0));
   for (int i = 1; i < nums.size(); i++) {
      int loc = i-1;
      while (loc >= 0 && tmp.get(loc) > nums.get(i)) {
         loc--;
      }
      tmp.add(loc+1, nums.get(i));
   }
   return tmp;
}
```

Specification of the helper methods used above:

Part A [5]. Suppose we call `foo` from another static method in the same class as follows:

```
myAL = foo(myAL);     // x
```

such that before the call, `myAL` is an `ArrayList` with the values `[7, 3, 5, 4, 9]`

What is the value of `myAL` after executing the statement labeled `x`:

Part B [2]. What is the worst case time complexity of `foo` as a function of the size of its parameter `nums` (use big-O notation):

Problem 3 [5 points]

Suppose we have the `ImPoint` class from assignment 1, but modified as shown below (new parts in bold):

```
public class ImPoint {
   private int x;
   private int y;
   private static final int ALPHA = 31;
   public  ImPoint(int x, int y) { this.x = x; this.y = y; }
   public boolean equals(Object other) {
      return this == other;
   }
   public int hashCode() {
      return ALPHA * x + y;
   }
   .  .  .  [other methods not shown]
}
```

Part A. Does the `ImPoint` class above satisfy the contract for `equals` and `hashCode` (contract shown below)? Assume `a` and `b` are both type `ImPoint`.

if `a.equals(b)` then `a.hashCode() == b.hashCode()`

Part B. Consider the following code using this version of `ImPoint` in a `HashSet` (a reminder about the `Set` interface is on the code handout):

```
Set<ImPoint> locs = new HashSet<ImPoint>();
ImPoint mc = new ImPoint(3, 4);
locs.add(mc);
ImPoint currentLoc = new ImPoint(3, 4);
boolean isThere = locs.contains(currentLoc);  // X
boolean added = locs.add(currentLoc);         // Y
boolean removed = locs.remove(currentLoc);    // Z
```

In the statement labeled `X` will `contains` look for `currentLoc` in the correct hash bucket (i.e., the one that the existing entry, `mc`, is in)?

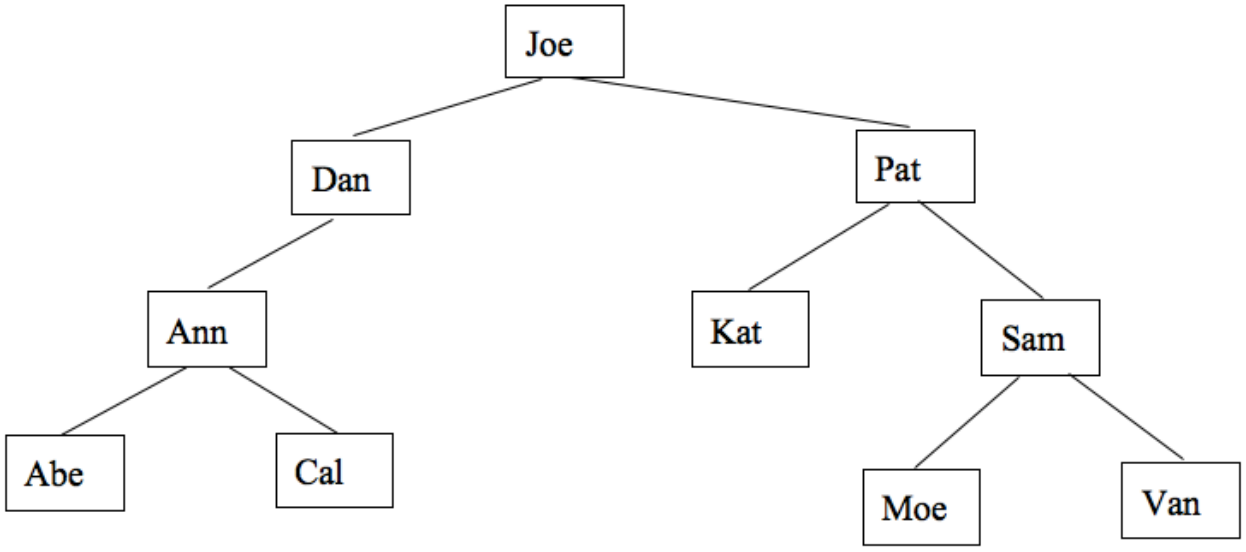
What is the value of `isThere` after statement `X` executes:

What is the value of `added` after statement `Y` executes:

What is the value of `removed` after statement `Z` executes:

Problem 4 [5 pts. total]

Binary search will **not** always work on the following tree (the root is shown at the top):



Give an example target search key such that a binary search on the given tree does NOT work correctly.

Part A. sample target search key:

Part B. sequence of keys your target key would be compared to in the attempted search:

Part C. result of the search on your sample search key: (true or false)

Problem 5 [5 points]

Suppose an exception can occur in a method you are writing. Which of the following are possible reasons why you might throw this exception from the method, i.e., using `throws` keyword in the method header, rather than catching the exception inside of the method? (circle the letter to the left of each that applies)

a. This method isn't responsible for the recovery policy for exceptions for this program.

b. If it's a checked exception, Java requires that we use `throws`.

c. The `try-catch` statement around the code that generates this exception already has a `catch` clause for this exception type.

d. The method has a non-`void` return type and there is no valid return value we could return from this method once this exception occurs.

e. If we don't throw the exception there is no way to let the user know there is a problem.

Problem 6 [15 points]

A different way to represent the maze data for the `Maze` class of assignment 3 is to just store the set of wall locations in the maze. In this problem you are going to write part of the implementation of the `Maze` class using this representation (using a Java `HashSet` – the instance variable is already defined for you below). **Implement (1) the `initWalls` private method (called from the constructor) and (2) the `getValAt` method.** For the purposes of this problem, assume `MazeCoord` has been defined appropriately to be able to use it as the element type in a `HashSet`. See code handout for more about the `MazeCoord` class.

```
public class Maze {

   private Set<MazeCoord> walls;  // the set of coords that are walls

   public static final int FREE = 0;
   public static final int WALL = 1;

   . . .  // other constants and instance variables not shown

   /**
     Constructs a maze.
     @param mazeData the maze data.  A 2D array of values Maze.FREE and
              Maze.WALL (see public FREE / WALL constants above) where the
              first index indicates the row.  e.g., mazeData[row][col]
   */
   public Maze(int[][] mazeData) {
      initWalls(mazeData);
      . . . [code to initialize the other instance variables not shown]
   }


   /**
     Initializes the walls in the maze.
     @param mazeData the maze data.  A 2D array of values FREE and
              WALL (see public FREE / WALL constants above) where the
              first index indicates the row.  e.g., mazeData[row][col]
   */
   private void initWalls(int[][] mazeData)  {  . . . }


   /**
     Gets the maze data at loc.
     @param loc the location in maze coordinates
     @return the value at that location.  One of Maze.FREE and Maze.WALL
     PRE: 0 <= loc.getRow() < numRows() and 0 <= loc.getCol() < numCols()
   */
   public int getValAt(MazeCoord loc) {  . . . }

   . . . [other parts of the class not shown]
}
```

[space for your answer on the next page]

Problem 6 (cont.)

```
public class Maze {
   private Set<MazeCoord> walls;  // the set of coords that are walls

   public static final int FREE = 0;
   public static final int WALL = 1;

   . . .  // other parts of the class not shown

   /**
     Initializes the walls in the maze.
     @param mazeData the maze data.  A 2D array of values FREE and
              WALL (see public FREE / WALL constants above) where the
              first index indicates the row.  e.g., mazeData[row][col]
   */
   private void initWalls(int[][] mazeData)  {









   /**
     Gets the maze data at loc.
     @param loc the location in maze coordinates
     @return the value at that location.  One of Maze.FREE and Maze.WALL
     PRE: 0 <= loc.getRow() < numRows() and 0 <= loc.getCol() < numCols()
   */
   public int getValAt(MazeCoord loc) {  . . . }
```

Problem 7 [17 points]

Write the static method `max`, which *recursively* computes the maximum value in an array of one or more `int`'s. 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 (array shown as comma-separated with brackets):

nums	*return value of call to* ***max(nums)***
`[4,5,3]`	`5`
`[-4,-2,-10,-2]`	`-2`
`[2]`	`2`

Little or no credit will be given for a solution that does not use recursion. For full credit the time complexity of your solution must be O(*n*) (where *n* is the length of the array).

```
// computes the maximum value in nums array
// PRE: nums.length >= 1
public static int max(int[] nums) {
```