The mantra of learning any language (programming language as well as real language) is *practice.*

We found that while most students had a reasonable high-level understanding of the concepts taught in the class, there was a big gap in their ability to implement them in code. One needs to practice writing and testing Java programs in Eclipse (or DrJava).

In the lectures, we gave homeworks that were not to be handed in (e.g. Piazza documents and additional material with the lectures), asking you to study various parts of the course coding guidelines and learn about loops, data structures, and techniques such as recursion. The recitations covered other things such as Strings in Java. But the prelim showed that many students did not practice.

So, this assignment is designed to make sure that you get that practice.

The assignment will be graded only on whether you turned it in and did a substantial amount of it. It is **an all-or-nothing grade**. Further, if you believe you know part of the material so well that you don't think you have to do the exercises, you can write,

"I know this material well enough so that I don't have to do this problem (or set of problems)."

Thus, it is your choice to do what you think is necessary, but *we will have a record of what you thought you knew*.

To repeat: if you believe you don't have to do a problem because you know the material already, just write something like "I know this material and don't have to do this problem".

When you write the methods, please strive for simplicity, brevity, clarity. Look at the development of the recursive function from the test on the slides for lecture 14 and the development of the string problem on the slides for lecture 15. Use the conditional expression where it leads to a simpler solution. Once you complete a solution, go over it again and simplify it, format it better, etc. Like writing an essay in English, writing a program requires writing and rewriting.

If you have to write a method for which we have not given a javadoc specification, write that specification FIRST, before writing the method. Make sure to mention all parameters by name in the specification.

The skeleton we give you has classes: TreeNode, E. We suggest that you create a JUnit testing class, Tester, and test all the methods you write. You will submit only TreeNode and E and a third document a5proofs, as discussed below

There are problems to do in these categories:

1. Strings and loops
2. Recursion, especially on trees
3. Algorithmic complexity
4. Induction

String-loop questions.

These functions are to be written in class E (for Exercises). Do not use recursion for these functions. Some of them require *you* to write the specifications and function headers, because you need practice with that. For these problems, you must be fluent with functions charAt(…), substring(…), indexOf(…), lastIndexOf(…), trim(), etc.

1. Write a function number(c, s) that returns the number of times character c appears in string s.
2. Write a function dupD that returns a copy of its parameter s but with all instance of 'd' duplicated. For example, dupD("adopted") equals "addoptedd".
3. String s contains a sequence of words (letters, upper case or lower case). Adjacent words are separated by one or more blanks, and there may be blanks at the beginning and end of s. Write a function fixCap that returns a copy of s with: (1) no blanks at the beginning and end, (2) exactly one blank between adjacent words, and (2) all words capitalized.

4. String s contains a sequence of first-and-last names separated by ";" and perhaps blanks after each ";" . Blanks may appear at the beginning and end of s. One blank separates each first and last name. Write a function fixNames that returns s with the same names but in the form "last, first" and with only 1 blank after each ";". For example, if s is " David Gries; Ashutosh Saxena;Haym Hirsh ", return the string "Gries, David; Saxena, Ashutosh; Hirsh, Hyam".
5. String s contains a time of day in 24-hour time. Examples: "0.5", "1:20", "12:50". "23:59". There is always a ":" in the string to separate hour and minutes. Write a function make12hr that returns time s in 12-our time. Thus, the above stings s are changed into: "00.05AM", "01:20AM", "00:50PM". "11:59PM". In the answer, the hour and minute should appear as two digits, with a preceding 0 if necessary.
6. Write a function fixA that returns its parameter s but with "_" inserted before and after each occurrence of 'a'. For example, if s is "anna", the answer is "_a_nn_a_".

Recursion questions

Put answers to all your recursion questions in class TreeNode at the place indicated. Note that we have given a recursive function to print binary trees.

1. Write a static function and a non-static function to return the number of nodes in a binary tree (These are stubbed in for you).
2. Write a static function and a non-static function to return the number of leaves of a binary tree.
3. Write a recursive function to computer b to the power c, for b a double and c an int that is >= 0.It should work in time O(c). Use the properties b**0 = 1, b**c = b*b**(c-1) and, for c even, b**c = (b*b)**(c/2).
4. (a) Write a boolean function isXish that, given two words, returns true if all the letters of the first word x are contained in the second s, *but in the same order*. For example, isXish("ab", "abc") is true but isXish("ba", "abc") is false. It is best to do recursion based on the first parameter, x.

   (b) A word is considered *elfish* if it contains the letters: e, l, and f in it, in that order. For example, the following words are elfish: whiteleaf, tasteful, unfriendly, and waffles, because they each contain those letters. Write a boolean function called isElfish that, given a word, tells us if the word is elfish or not. The difference between (a) and (b) is that (a) requires order while (b) does not.

5. Write a function permutations that, given a string s of all-different characters, returns a Set<String> that contains all permutations of s. This function is stubbed in class TreeNode with a suitable specification. Here are some hints. (1) Set is an interface, so you cannot create an instance of it. Instead, create an instance of class HashSet. (2) The base case is s empty or s consisting of 1 character. (3) In the recursive case, the answer is the set of all permutations of s[1..] modified to insert s[0] at each possible place. Example. If one permutation of s[1..] is "bdc" and s[0] is 'a', then these permutations belong in the answer: "abdc". "badc", "bdac", "bdca".
6. [Optional] Coin game: Alice and Bob are playing a game using a bunch of coins. The players pick several coins out of the bunch in turn. Each time a player is allowed to pick 1, 2 or 4 coins. The player that gets the last coin is the winner. Assume that both players are very smart and will try their best to work out a strategy to win the game. For example, if there are 2 coins and Alice is the first player to pick, she will definitely pick 2 coins and win. If there are 3 coins and Alice is still the first player to pick, no matter whether she picks 1 or 2 coins, Bob will get the last coin and win the game.

   Given the number of coins and the order of players (which means the first and the second players to pick the coins), write a method to calculate the winner of the game and calculate how many different strategies there are for them to win the game. Use recursion to solve the problem. You can assume that there are no more than 30 coins.

   Here are some sample runs of the program:

pickCoin(1, "", ""): Alice winds, with 1 strategy
pickCoin(2, "Bob", "Alice"): Bob winds, with 1 strategy
pickCoin(3, "Alice, "Bob"): Bob wins, with 2 strategies
pickCoin(10, "Alice", "Bob"): Alice wins, with 22 strategies
pickCoin(35, "Alice", "Bob"): Alice wins, with 3344 strategies
pickCoins(30, "Alice", "Bob"): Bob wins, with 18272 strategies

Answer the asymptotic complexity questions and the induction problems in a Word document or txt file named a5proofs (with suitable extension).

Asymptotic complexity problems

1. Prove formally that function g(n) = 2n+2 is O(n).
2. Prove formally that function g(n) = 2n + 2 is $O(n^2)$. Yes, this seems to contradict exercise 1. Give an explanation for this.
3. Prove formally that function h(n) = $7n^2$ + 2n + 1000 is $O(n^2)$.
4. Prove formally that function k(n) = $10^n + n^2$ is $O(2^n)$
5. Below is a static function find(String x, String e). Write down its worst-case order of execution O(|x|, |s|), where the notation |x| denotes the length of x. Explain why that is the runtime of this method. Also explain what you think its expected order of execution is.

```
/** Return the index of the first occurrence of x in s (-1 f it is not in) */
public static int find(String x, String s) {
   int h= x.length();
      for (int k= 0; k < s.length(); k= k+1) {
           if (x.equals(s.substring(k, k+h))) {
              return k;
           }
      }
      return -1;
}
```

Induction problems

Answer these questions in comments at the appropriate place in file E.java. You can wait to do this until we have a lecture on induction.

1. Prove that for n >= 0, P(n) holds. P(n): 1 + 3 + 5 + … + 2n-1 = n^2.
2. Prove that for n >= 0, P(n) holds: P(n): 1 + 2 + 3 + … + n = n(n+1)/2.
3. Prove that for any n > 0, P(n) holds: P(n): n^3 + 2n is divisible by 3.
4. Prove that for any n >= 1, P(n) holds: P(n): $3^n > n^2$.
5. In TreeNode, you wrote static function size(TreeNode r). Prove by induction on the size of r that the function satisfies its specification.