CS 142 – Spring 2015 **Name: ________________________________**

Recursion Practice (10 quiz points)

1. Implement a recursive binary search function, using what we learned in class on Wednesday.
   Write your function below.

2. What is the run-time of the recursive binary search? ________________________________
   Hint: it's the same as iterative binary search.

3. Modify the **`power_to`** function you wrote last Friday to make it more efficient
   a. Think about the symmetry of the problem
      i. If b is even, $a^b = a^{b/2} * a^{b/2}$
      ii. If b is odd, $a^b = a^{b/2} * a^{b/2} * a$
   b. Therefore, we don't need to compute $a^{b/2}$ twice

   Write your function below.

4. The run time of the original **`power_to`** function we wrote last Friday shown below is O(n) where n = b.

```
def power_to(a, b):
    if b == 1:
        return a
    else:
        return a * power_to(a, b-1)
```

   What is the run-time of the **`power_to`** function you wrote for question 3? ____________________________
   Hint: think about how it relates to the binary search algorithm.

5. **Challenge Problem (worth up to an extra 5 quiz points):**

Write a recursive function to find the edit distance between two strings. The objective is to find the least number of substitutions, insertions, and deletions required to get from one string to another.

Examples: `distance(‘spam’, ‘poems’) returns 4`

`distance(‘alien’, ‘sales’) returns 3`

```
def distance(first, second):
```

Base cases:

1. If first is the empty string, return the length of second
2. If second is the empty string, return the length of first

Recursive Cases:

Consider each of the three options.

1. If we are going to perform a substitution, you can see that we should change the first letter in first to be the same as the first letter in second. Now these letters will match and we can remove those two letters from further consideration. Therefore, the best solution that begins with a substitution will have cost 1 + distance(first[1:], second[1:]) because the number of operations will be 1 (the substitution) plus however many operations are required to get from the remaining string first[1:] to the remaining string second[1:].

2. The second option is deleting the first symbol in first. That’s one operation, and now we would need to find the distance from first[1:] to second.

3. Lastly, we need to consider inserting a new symbol to the front of first. You can see that the new symbol should match the first symbol in second. That will require one operation, and then the remaining problem is to find the distance from first to second[1:]. That’s because we haven’t yet used the first symbol in first, but we’ve just matched the first symbol in second.

The best of these three options will be our best solution!

Write your function below; if you don’t get this completed by the end of class, you may hand it in on Monday, or write down what you do have working for possible partial credit.