Name(s)_____ _____

Algorithms, Assignment 1: Loop Invariants, Asymptotic Notation, and Running Time

1. Prove correctness of the following Python code using a proof by loop invariant.

```
def find_closest(array, val):
    '''Find the value that is the closest to the given val.'''

    i = 0
    n = len(array)
    closest_val = None
    closest_dist = infinity

    while i < n:
        current_val = array[i]
        current_dist = abs(array[i] - val)

        if current_dist < closest_dist:
            closest_val = current_val
            closest_dist = current_dist

        i += 1

    return closest_val
```

(a) Loop invariant (write a statement that can be easily proven true or false, that references the purpose of the loop, and references variables that change each iteration):

(b) Initialization (show that the loop invariant is true before the loop starts):

Name(s)_____ _____

(c) Maintenance (show that the loop invariant holds when executing any iteration):

(d) Termination (show that the loop invariant holds once the loop ends):

Name(s)_____ _____

2. Show that for any real constants a and b, where $b > 0$,

$$(n + a)^b = \Theta(n^b)$$

Note: the problem is asking you to do a **Theta** proof, not just Big-O.

Name(s)_____ _____

3. Consider the following problem:

Search five separate arrays (sequentially) for a given number.

(a) Write code to solve the problem. You may use any programming language or any form of pseudocode.

(b) What is the **total running time** of your code? _____

(c) What is the **asymptotic running time** of your code? _____

(d) Prove that your answer to (b) is Big-O of (upper bounded by) your answer to (c). (You must find appropriate values for c and n_0.)

Name(s)_____ _____

4. Fill in the following table with the running times that you had as output of your Insertion Sort program.

	Insertion Sort	Merge Sort
random10.txt		
random1000.txt		
random10000.txt		
increasing10.txt		
increasing1000.txt		
increasing10000.txt		
decreasing10.txt		
decreasing1000.txt		
decreasing10000.txt		

(a) Write down **two** observations you can make from the table above.

(b) Look at the code and see how timing is performed (look for "`timer`" in the given code). How could our code timing (sometimes called profiling) be improved?