Name ______________________________

Problem 1 (20 points (10 each))

The following are proposed proofs of the given claims. Below each one, write whether the proof is valid or not. If it is *not* valid, explain why. Only a sentence or so should be needed to explain why a proof does not work.

(a) *Claim.* $n^2 - 56$ is not $O(n)$.

Proof. In order to show that $n^2 - 56$ is not $O(n)$, we need to show that there do not exist real numbers $c > 0$, $n_0 \geq 0$ such that $\forall n \geq n_0 : n^2 - 56 \leq c \cdot n$. Notice that $n^2 - 56$ and n intersect at $n = 8$, so consider $n_0 = 9$ and $c = 1$. When $n = 10$, $n^2 - 56 = 100 - 56 = 44$, which is greater than 10. So $\forall n \geq 9 : n^2 - 56 \leq 1 \cdot n$ does not hold, so $n^2 - 56$ is not $O(n)$. □

(b) *Claim.* The best-case runtime of recursive binary search (below) is $O(1)$.

Algorithm 1 `binarySearch`(A[1...n], x)

```
if $|A| = 0$ then
    return False
else
    $middle = \lfloor \frac{|A|}{2} \rfloor$
    if $A[middle] = x$ then
        return True
    else if $A[middle] > x$ then
        binarySearch(A[1..$middle - 1$], x)
    else
        binarySearch(A[$middle + 1$...1], x)
```

Proof. Note that the `if` statement on lines 1-2 uses a constant c number of primitive operations and that the `else` on lines 3-10 uses a constant d operations. In the best case, $|A|$ is 0, so only the `if` statement is executed and the runtime is $f(n) = c$. Since any degree k polynomial is $O(n^k)$ and $f(n) = c$ is a degree zero polynomial, this algorithm has a best-case runtime of $O(1)$.

□

Problem 2 (20 points)

In this problem, you will prove that $2n^2 + 3 = O(n^3)$ using the definition of big O. Follow the three steps carefully.

(5 points) Write down the definition of big O.

(10 points) Give a c and a n_0 that can be used to prove that $2n^2 + 3 = O(n^3)$.

(5 points) Explain what it would mean for this c and n_0 to work in a proof that $2n^2 + 3 = O(n^3)$, and *very briefly* explain why they do (write one sentence, draw a graph, etc).

Problem 3 (20 points)

(a) For the following algorithm give a proposed function representing the number of primitive operations for the algorithm in terms of the input size, addressing each line and/or loop of the algorithm. You do not need to be precise counting constant numbers of primitive operations (e.g., figuring out exactly how many primitive operations a single line does). However, you should try to be precise about how many times a loop runs.

Algorithm 2

```
1: for $i = 1$ to $2n$ do
2:     $j = n$;
3:     while $j > 1$ do
4:         $j = j/3$;
```

Algorithm 1 takes $f(n) =$ primitive operations.

(b) For the $f(n)$ you gave in (a), give the "tightest" (aka asymptotically smallest) $g(n)$ such that $f(n) = O(g(n))$ (That is, give g such that $f(n) = \Theta(g)$.

$f(n) = O(\quad)$

Problem 4 (20 points)

Recall the recursive binary search algorithm from Problem 1:

Algorithm 3 `binarySearch`(A[1...n], x)

```
if $|A| = 0$ then
    return False
else
    $middle = \lfloor \frac{|A|}{2} \rfloor$
    if $A[middle] = x$ then
        return True
    else if $A[middle] > x$ then
        binarySearch(A[1..$middle - 1$], x)
    else
        binarySearch(A[$middle + 1$...1], x)
```

Notice that on line 4, `binarySearch` calculates *middle* as $\lfloor \frac{|A|}{2} \rfloor$, meaning that if $\frac{|A|}{2}$ is non-integer, it is rounded down to the nearest integer.. To give a rigorous analysis of the worst-case runtime of `binarySearch`, we should account for the fact that the algorithm behaves slightly differently when n is odd versus when n is even.

In this problem you will give a recurrence relation for the worst-case runtime of `binarySearch`. You may assume that the `if` statement on lines 1-2 uses a constant c number of primitive operations, and that the `else` on lines 3-10 uses a constant d operations.

(a) Describe the worst-case input for `binarySearch` on an array of size n.

(b) Give the base case of the recurrence relation. Make sure you use the smallest possible input to the algorithm.

(c) Give the recursive case of the recurrence relation. Hint: your recurrence relation should address both the case where n is even and n is odd. That is, fill in the blanks below.

$$T(n) = \begin{cases} \qquad\qquad\qquad\qquad , \text{if } n \text{ is even} \\ \qquad\qquad\qquad\qquad , \text{if } n \text{ is odd} \end{cases}$$

Problem 5 (20 points)

Each is worth 5 points.

(a) True or false: if $f(n) = \begin{cases} 6n^2 + 2n + 3 & \text{if } n \text{ is odd,} \\ 6 & \text{otherwise} \end{cases}$, then $f(n) = O(n^2)$.

(b) Recall that big O forms a relation on the set of functions $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ where function f_1 is related to function f_2 if $f_1 = O(f_2)$. The claim "the big O relation on the set of functions $f : \mathbb{R}^{\geq 0} \to \mathbb{R}^{\geq 0}$ is comparable" is false. Recalling that a relation R on set S is comparable if $\forall x, y \in S : xRy$ or yRx, how would you go about disproving the claim? (You don't need to actually disprove it here, just explain how you would.)

(c) Give a disproof to the claim from (b).

(d) Iterate the following recurrence relation for $n = 0$ through $n = 4$: $T(0) = 3$; $T(n) = T(n-1) + n$.

- $T(0) =$
- $T(1) =$
- $T(2) =$
- $T(3) =$
- $T(4) =$

(e) What is $\log_2 16$?