Problem 1 (Drop It)

(a) Implement a Racket function `dropzeroes` that takes a list `l` of numbers as input and returns `l` but with all 0s. For example, `(dropzeroes '(1 3 0 1 0 2))` --> `(1 3 1 2)`. Make sure to test your code in DrRacket and add your code to this write-up using `lstlisting`.

(Hint: This function is like `remove`, but specialized to 0s.)

(a) Prove the following property of `dropzeroes`:

**Claim 1 (Idempotence of Dropzeroes).**
\[
(dropzeroes (dropzeroes l)) = (dropzeroes l)
\]

You may take the shortcut of evaluating a recursive function call directly to the branch that it selects in a single step.

(Hint: You’ll need to perform multiple case analyses in this proof. Be diligent and precise in your evaluation and what facts you learn from the case analyses.)
Problem 2 (Natty)

Consider the following Racket definition

```
(define pow
  (lambda (x y)
    (if (zero? y)
        1
        (* x (pow x (- y 1))))))
```

Prove the following claim using mathematical induction. You may take the shortcut of evaluating a recursive function call directly to the branch that it selects in a single step.

Claim 2. For all natural numbers x, y, and z, \((\ast (\text{pow} \ x \ y) \ (\text{pow} \ x \ z)) \equiv (\text{pow} \ x \ (+ y z))\).

In your proof, you can assume that the following lemma about pow holds:

Lemma 1. For all natural numbers x and y, \((\ast \ x \ (\text{pow} \ x \ y)) \equiv (\text{pow} \ x \ (+ \ y \ 1))\).

In your proof, make sure to be explicit where you invoke both this lemma and your induction hypothesis.
Problem 3 (Moving On Up)
Consider the following recursive Racket definitions:

```racket
(define even?  
  (lambda (n)  
    (cond [(zero? n) #t]  
          [(= n 1) #f]  
          [else (even? (- n 2))])))

(define iterate  
  (lambda (f x n)  
    (if (zero? n)  
        x  
        (iterate f (f x) (- n 1)))))
```

For this problem, you may take the shortcut of evaluating a recursive function call directly to the branch that it selects in a single step.

(a) First prove the following auxiliary claim about `even?`:

**Claim 3.** For any natural number `n`, if `(even? n)` and `n` is not zero then `(<= 2 n)`.

(b) Consider the following claim about `iterate`:

**Claim 4.** For all natural numbers `n` and booleans `b`, if `(even? n)` then `(iterate not b n) ≡ b`.

In a sentence or two, describe at a high-level why this claim is correct.

(c) Formally prove this claim, using the auxiliary claim from part (a).

(Hint: Because of what evenness tell you, you will likely need to consider strong induction to “get to” the next even number in your analysis.)

(Hint: Be precise about your induction hypothesis. If you do it right, you will need to first to prove a preliminary fact before you can use your induction hypothesis.)
Problem 4 (Proof Busters)

Consider the following inductive proof of the given claim.

Claim 5. For all natural numbers \( n \), \( n + 1 \leq n \).

Proof. Proof by induction on \( n \). In the inductive case where \( n = k + 1 \), our induction hypothesis states that \( k + 1 \leq k \). We must then show that \( (k + 1) + 1 \leq k + 1 \):

\[
\begin{align*}
(k + 1) &\leq k & \text{inductive hypothesis} \\
(k + 1) + 1 &\leq k + 1 & \text{transitivity of } \leq
\end{align*}
\]

and thus our goal is immediately proven.

This claim is clearly bogus. In a few sentences, describe the mistake(s) in logical reasoning that this proof commits and how the falsity of each claim breaks the proof in each mistake you identify.

(Hint: think carefully about (a) the structure of inductive proof and (b) the assumptions that each step of the proofs are making.)