```
CPSC320 Assignment 2
====================
--------------------
|Name:  Evan Louie |
|CSID:  m6d7       |
|SID:   72210099   |
--------------------
=======
1.
=======

  The population of the coins follow a binomial distribution; meaning each coin has two
  possiblilities in state:
    i)  Real
    ii) Fake
  This means that for n coins their exists 2n possible answers in which a single coin is a fake
  and 2n+1 possibe answers in which the possiblity that all are real exist.

  We can also show this in terms of a decision tree in which, given n coins, each coin has the
  option to be lighter or heavier. Because their are n leaves (coins) which can be lighter, or
  n leaves to be heavier or one case in which they are all real; The amount of leaves is 2n+1

  Because the decision tree is a ternary tree with 3 possiblities at each branch (heavier,
  lighter, equal), the height of the tree, h, can be used to calculate the maximum number of
  leaves the tree has by evaluating 3^h (by definition of a ternary tree).

  This means that 3^h ≥ 2n+1
  If we solve for h:
    h ≥ log(3, 2n+1)
    h ≥ ceiling(log(3, 2n+1)) (because h is an integer)
    [read: ceiling of log base 3 of 2n+1]

    Therefore the algorithm must perform at least ceiling(log(3, 2n+1)) weighings.

  QED



=======
2.
=======
  -----
  (a)
  -----

    TRUE;

    Definition Of Lower Bound Ω:
    ----------------------------
      Let T(n), T : N → R+, be a function that describes the worst-case running time
      of a given algorithm in terms of steps to be completed per size of input n.
      We say that T(n) is Ω(f(n)) i.e. we say that (T(n) is asymptotically lower bounded
      by f(n), f : N → R+, i.e. if there is a constant c ∈ R+ and n0 ∈ N such that
      T(n) ≥ c · f(n) for ∀n ∈ N, n ≥ n0. If T(n) is Ω(f(n)) we also write T(n) = Ω(f(n))
```

Using This definition, then it logically follows that if $f \in \Omega(g)$ is equivelant to $f(n) \geq c \cdot g(n)$ for some c and $n \geq n0$.

We can first prove that $f \in \Omega(g)$:

Premise 1: $f(n) \geq c \cdot g(n)$

Let:

$f(n) = n$
$g(n) = n$
$c < c$
$n0 = 1$

If we then evaluate the function in Premise 1:

$n \geq (c) \cdot (n) \quad \forall n \in N,\ n \geq n0$

Which holds true and can be proven by taking the first derivative of each side of the function

$d/dn(n) = 1$
$d/dn(nc) = c$

As the first derivative of n is greater than or equal to nc when $0 \leq c \leq 1$ and the second derivative of both is zero:

We have shown $f \in \Omega(g)$ and Premise 1 holds.

Now we can prove that $f^2 \in \Omega(g^2)$:

Since we have proven Premise 1, it is a trivial matter to prove that applying any exponential growth function to either side will lead to the same result by applying the same method of proving as for Premise 1:

Let:

$f(n) = n$
$g(n) = n$
$c = c$
$n0 = 1$

If we evalaute the function:

$(f(n))^2 \geq c \cdot (g(n))^2 \quad \forall n \in N,\ n \geq n0$
$(n)^2 \geq (c) \cdot (n)^2 \quad \forall n \in N,\ n \geq n0$

The function holds true and can be proven by taking the first and second derivatives of each side.

derivative$(n^2) = 2n$
derivative$(2n) = 2$

derivative$(c(n)^2) = 2(c)(n)$
derivative$(2cn) = 2(c)$

As any value $c \geq 1$ will make the rate of the rate of growth of $c(n)^2$ greater than that of n^2,

$(f(n))^2 \geq c \cdot (g(n))^2 \quad \forall n \in N,\ n \geq n0$

holds true, we have proven that if $f \in \Omega(g)$ is true, $f^2 \in \Omega(g^2)$ is also TRUE.

QED

(b)

FALSE;

Definition of Upper Bound O:

Let $T(n)$, $T : N \rightarrow R+$, be the function that describes the worst-case running time of a given algorithm in terms of steps to be completed per sitesize of input n. We say that T(n) is O(f(n)) (read ”T(n) is of order f(n)”) if T is asymptotically upper bounded by f(n), f : N → R+, i.e. if there is a constant c ∈ R+ and n0 ∈ N such that T(n) ≤ c · f(n) for ∀n ∈ N, n ≥ n0. If T(n) is O(f(n)), we also write T(n) = I(f(n)).

Using the definition of Upper Bound O, f ∈ O(g) is equivelant to f(n) ≤ c·g(n) for some c and n≥n0.
This means that g(n) is not necessarily a member of f(n) however a constant c exists that when multipied together with g(n) makes it so.
With this is mind, we can make an example in which:

f(n) = 2n
g(n) = n

We know that:

some c exists such that f(n) ≤ c·g(n)

```
Example:  if (c == 3) {
              f(n) < c·g(n)
          }
```

However this property does not transfer to a function which applies f(n) and g(n) as an exponential value; such as the one asked here;
If we define f(n) and g(n) using the same values as they were ealier when showing that f(n) ≤ c·g(n) we get:

f(n) = 2n
g(n) = n

This means that:

2^(f(n)) = 2^(2n)
2^(g(n)) = 2^(n)

Are question asks to prove/disprove 2^f ∈ O(2^g), so mathimatically speaking, for some c:

2^f(n) ≤ c·2^g(n)
2^(2n) ≤ c·2^(n)

Must be true for all values n greather than some n0.
To state this otherwise is:

Limit(2^(2n), infinity) ≤ Limit(2^(n), infinity)
[Read: Limit of 2 to the power of 2n is lesser than or equal to the limit of 2 to the power of n when n approaches infinity]

We can show this is never true as taking the second derivative of both evaluate to:

derivative(derivative(2^(2 n)) = 4^(n+1)·log^2(2)
derivative(derivative(2^n)) = 2^n·log^2(2)

As shown, the rate which the rate of 2^(2n) increases is greater than the rate of the rate of 2^(n), therefore, no c exists such that:

2^(2n) ≤ c·2^(n)

can be true for some n≥n0

We have proved that even if f ∈ O(g) is true, 2^(f) ∈ O(2^g) is FALSE.

QED.