1. (a) Show that \( \log_2 x = O(\log_4 x) \).
(b) Show that if \( a \) and \( b \) are real numbers with \( a > 1 \) and \( b > 1 \), if \( f(x) = O(\log_a(x)) \), then \( f(x) = O(\log_b(x)) \).

2. Find a \( O(n) \) estimate for the run-time of the piece of code given below.

```plaintext
m:=0
for (i:=1 to n)
    for (j:= i * i to n * n)
        m:=i + j
```

3. There is a more efficient algorithm (in terms of the number of multiplications and additions) for evaluating polynomials than the one we considered in worksheet 7. It is called Horner's method. Consider the following pseudocode for this procedure, which finds a solution to the polynomial \( a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \) at \( x = c \):

```plaintext
procedure Horner(c, a_0, a_1, \ldots, a_n)
    y = a_n
    for i := 1 to n
        y := y * c + a_{n-i}
    return y
```

(a) Evaluate \( x^4 - 4x^3 + 2x^2 + x + 3 \) at \( x = 2 \) by working through each step of the algorithm and showing the values assigned at each step. (Make sure to write EVERY value a variable gets if it changes during the algorithm.)

(b) Exactly how many multiplications and additions are used by this algorithm to evaluate a polynomial of degree \( n \) at \( x = c \)? (You don’t need to count additions used to increment \( i \) in the for loop.)

4. Devise an algorithm that finds all terms of a finite sequence \( a_1, \ldots, a_n \) of positive integers that are greater than the sum of all the previous terms of the sequence. Analyze the complexity (number of comparisons, additions, and multiplications) of your algorithm.
   Note: Algorithms with better complexity/runtime will be given more credit!

5. Extra credit: Suppose that \( f(x) = O(g(x)) \). Does it follow that \( 2f(x) = O(2g(x)) \)? Prove your answer.