Homework: Recurrences

Name: ________________________________

1. Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$. (exercise 4.3–1, p. 87)
   (a) (5 points) Show the solution using the substitution method.

   (b) (5 points) Show the solution using the recursion tree method.

2. (10 points) Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(\frac{n}{2}) + n^2$. Use the substitution method to verify your answer. (Exercise 4.4–2, p. 92)

3. (5 points) Recall the Fibonacci sequence, $\{1, 1, 2, 3, 5, 8, 13, 21, \ldots\}$. The following code snippet is a recursive implementation of the Fibonacci sequence:

```
function fibonacci(n)
  raise error if n < 0
  if n = 0 or n = 1 then // or n < 2
    return 1
  else
    return fibonacci(n - 1) + fibonacci(n - 2)
  end
end
```

The recurrence relation, $f(n) = f(n-1) + f(n-2), n > 1$ describes how to compute the Fibonacci number at location n. Sketch a recurrence tree for $f(n)$ with a depth of 4.

Notice how quickly this tree grows. In fact $T(n) = \Omega(c^n)$. Where c is $\frac{1 \pm \sqrt{5}}{2}$. It's also $O(2^n)$. How would we classify this time (i.e. constant, linear, etc.)?

4. (5 points) The following code snippet is an iterative implementation of the Fibonacci sequence:

```
function fibonacci(n)
  raise error if n < 0

  f0 = 1
  f1 = 1
  fn = 1
  for j in 2..n loop
    fn = f0 + f1
    f0 = f1
    f1 = fn
  end
  return fn
end
```

Analyze the running time in Big O notation for the iterative implemenetation.

Which is faster, the recursive or iterative Fibonacci implementation? Explain your answer.