Practice Exam 1

You will have 1 hour for this exam, although you should not need that long. This exam is closed-book and closed-note. Please take some time to check your work. If you need extra space, write on the back. You must show your work to receive any partial credit. There are a total of 25 points on this exam.

1. (8 points) Consider the following Scala function:

   def m(a: Int, b: Int): (Int, Int) = {
     var x = a
     var y = 0
     while (x >= b) {
       x = x - b
       y = y + 1
     }
     (y, x) // Return this pair
   }

   (a) What is the result of \(m(10, 3)\)?

   (b) Give an invariant relating the values of \(x\) and \(y\) each time the while test is evaluated:

   (c) What function is computed by \(m(a, b)\)? Support your claim using your invariant. You should assume that \(a \geq 0\) and \(b > 0\).
2. (5 points) Suppose the running time $T(N)$ of some algorithm is given by the following recurrence:

$$\begin{align*}
T(1) &= 1 \\
T(N) &= T(N - 1) + 2N - 1, \quad (N > 1)
\end{align*}$$

(a) Fill in the following table of values. For the last entry, give a closed-form expression for $T(N)$, either by solving the recurrence or by guessing:

<table>
<thead>
<tr>
<th>T(1)</th>
<th>T(2)</th>
<th>T(3)</th>
<th>T(4)</th>
<th>T(N)</th>
</tr>
</thead>
</table>

(b) Prove by induction that your closed-form expression for $T(N)$ is correct.
3. (12 points) Here is our Scala code for inserting a value in a binary search tree:

```scala
trait Tree
case object Empty extends Tree
case class Node(left: Tree, value: Int, right: Tree) extends Tree

def insert(t: Tree, n: Int): Tree = t match {
  case Empty => Node(Empty, n, Empty)
  case Node(l, v, r) =>
    if (n == v) // No change -- already in tree
      t
    else if (n < v)
      Node(insert(l, n), v, r)
    else // n > v
      Node(l, v, insert(r, n))
}
```

(a) Complete the following skeleton to define a function `insertAll` which takes a tree and a list of numbers and returns a new tree with all of the numbers inserted into the original tree:

```scala
def insertAll(t: Tree, nums: List[Int]): Tree = nums match {
  case Nil =>
    case head :: tail =>
}
```

(b) Show the tree which results from evaluating `insertAll(Empty, List(3, 1, 4, 1, 5))`:

(continued)
(c) Give a tight big-oh upper bound on the average running time of `insertAll` in terms of the size of the list, $N$ (assume that the initial tree is empty, and that the resulting tree is “balanced”):

(d) Here is a version of inorder traversal which returns the visited items in a list (the ::: operator concatenates two lists; assume for this problem that this can be done in constant time):

```scala
def inorder(t: Tree): List[Int] = t match {
  case Empty => Nil
  case Node(l, v, r) => inorder(l) ::: List(v) ::: inorder(r)
}
```

Now we may define the following function:

```scala
def doSomething(nums: List[Int]): List[Int] = inorder(insertAll(Empty, nums))
```

What is the result of `doSomething(List(3, 1, 4, 1, 5))`?

(e) Describe the effect of `doSomething(nums)` on an arbitrary list `nums` of type `List[Int]`:

(f) Give a tight big-oh upper bound on the average running time of `doSomething` in terms of the size of its argument, $N$: 