Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Spring 2004
Homework 4b
Due on 2/17/04 (beginning of class)

1. **[75 Points] Implementing B-Trees!** In this assignment you will implement B-Trees. In particular, your implementation should assume that $t = 2$. Your code should have a `Node` element similar to the one suggested in the lecture notes. Your program should support the operations `insert(x)`, which inserts the integer `x` into the tree, and `find(x)` which returns a boolean indicating whether or not `x` is in the tree. For insertion, use the one-phase method described in class (in which insertion is done in a single pass down the tree). In addition, your program should have a method called `height()` which returns the height of the tree. (A tree with a single node has height 0.) Please do this programming exercise without looking at the book. For one thing, the book's pseudocode isn't particularly clean or clear. In addition, this is the first chance you'll have had to write your own code given high-level ideas presented in class—I suggest you start by writing your own pseudo-code. If you're careful at the design stage, you might be able to get away with very little debugging.

   You should turn in a copy of your code and a sample run of a test program which inserts the even numbers from 0 to 20 (in order) into the tree. Next, invoke the `height()` method and print the height of your tree. Then, the test program tests to see if each number from 0 to 20 (odd and even) is in the tree and prints "true" or "false" for each such query.

   You have several options when turning in a sample run of your program:

   (a) Use script (do `man script` to learn more). Do not create such a script inside an emacs buffer because it won't print correctly! In fact, I don't recommend `script` because there are a variety of situations in which it won't print correctly.

   (b) Run your program in an emacs `shell` buffer interactively. Cut and paste the parts of the interaction that show your program running into another buffer, save as a text file, and then print this.

   (c) Once you're sure your program works the way you'd like, you could use redirection to generate text files.