Name: _____

Exercise D1, 4/5/05

Part 1: Big-*O* Notation

The big-*O* notation is useful for analyzing the effect of the input data size on the performance of an algorithm (not including the constant factor corresponding to, e.g., fixed amount of speed up). Although this concept must have been discussed in other courses, e.g., Advanced Algorithms, it would still be good to review the idea. To do so, we will re-use some of the tools introduced earlier in this course, i.e., game-theoretic interpretation of FOL and the on-line graphing tool. First, here is a definition of the big-*O* notation.

Definition: '*O*' (asymptotic upper bound) is defined as follows: $f(n) \in O(g(n))$ if there are constants $c > 0$ and $n_0 \geq 1$ such that $f(n) \leq c\, g(n)$ for every integer $n \geq n_0$.

Task: Give a game-theoretic interpretation of the above definition *referring to the operation of the on-line graphing tool* (http://www.tcnj.edu/~komagata/Graphing). Note that you can adjust the constant *c* using a slide bar (and a text field) and designate some n_0 by simply identifying a specific value on the *x* axis (note that you can change the graph scale).

Hint: If you are not sure how to use the tool, read the tutorial part of the documentation linked from the graphing tool page.

Part 2: Sample Problems

As discussed in class, there are a large number of NP problems. NP problems are important because many practical *combinatory* problems exhibit the properties of this class.

Task: Choose at least two problems from the Sample Problems (http://www.tcnj.edu/~komagata/csc460/05s/SampleProblems.pdf) listed below, and informally explain why the problems are in the class NP.

- #11 Cryptography
- #12 Monkey Puzzle
- #15 Knapsack Problem
- #17 CPU Register Allocation
- #18 Map Coloring

Survey: Time spent between classes: _____

// End