Midterm Exam, CMSC 420, Fall 2002 **Name:**_____

Instructions:

- Leave at least one empty seat between you and each of the other students.
- This exam is open book, open notes (and closed neighbor :-)
- Each question is worth 5 points, for a total of 100 points.
- If you want any partial credit for wrong answers, you need to show your work.
- Write only on the test sheets. If you run out of room, write on the back of the last page.

Problem 1. Professor Prune has created an algorithm that does some kind of computation on an array. If the algorithm is called on an array of size n, then the number of comparison operations done by the algorithm will vary depending on *which* array of size n, but it will always be a number in the interval from $2n^2 - n$ to $n^3 + 2n - 1$, inclusive.

1. Let $C_{\text{best}}(n)$ and $C_{\text{worst}}(n)$ be the worst-case and best-case values for the number of comparisons done by the algorithm. Below, circle each true statement:

$C_{\text{best}}(n) = O(n^2)$	$C_{\text{best}}(n) = \Omega(n^2)$	$C_{\text{best}}(n) = \Theta(n^2)$
$C_{\text{best}}(n) = O(n^2 \lg n)$	$C_{\text{best}}(n) = \Omega(n^2 \lg n)$	$C_{\text{best}}(n) = \Theta(n^2 \lg n)$
$C_{\text{best}}(n) = O(n^3)$	$C_{\text{best}}(n) = \Omega(n^3)$	$C_{\text{best}}(n) = \Theta(n^3)$
$C_{\text{worst}}(n) = O(n^2)$	$C_{\text{worst}}(n) = \Omega(n^2)$	$C_{\text{worst}}(n) = \Theta(n^2)$
$C_{\text{worst}}(n) = O(n^2 \lg n)$	$C_{\text{worst}}(n) = \Omega(n^2 \lg n)$	$C_{\text{worst}}(n) = \Theta(n^2 \lg n)$
$C_{\text{worst}}(n) = O(n^3)$	$C_{\text{worst}}(n) = \Omega(n^3)$	$C_{\text{worst}}(n) = \Theta(n^3)$

2. Let $I_{\text{worst}}(n)$ be the total number of computer operations (including both comparison operations and other operations) done by Professor Prune's algorithm in the worst case on an array of size n. Is it possible to have $I_{\text{worst}}(n) \neq O(C_{\text{worst}}(n))$? Explain why or why not.

Problem 2. A *tridiagonal* matrix is a square matrix $T[1..n, 1..n]$ in which $T[i, j]$ is nonzero only if $|i - j| \leq 1$. You are to develop a sequential storage allocation strategy for tridiagonal matrices that only allocates storage to the nonzero entries.

1. Draw a picture of the memory locations for a 4×4 tridiagonal matrix $T[1..4, 1..4]$. At each memory location, write a note saying which $T[i, j]$ it represents.

2. How many storage locations will be needed to represent the nonzero elements of a tridiagonal matrix $T[1..n, 1..n]$? (Give an exact value, not a Θ or big-O value).

3. Write a formula that gives the location of an element $T[i, j]$ of a tridiagonal matrix $T[1..n, 1..n]$.

4. How many operations does it take to calculate the location of an element $T[i, j]$ of a tridiagonal matrix $T[1..n, 1..n]$? (Give an exact value, not a Θ or big-O value).

5. Professor Prune says, "Instead of using sequential storage to represent a tridiagonal matrix, it's just as good to use the linked-list representation shown on page 140 of the book. In both cases the time needed to retrieve the value of an element $T[i, j]$ is $\Theta(1)$, and in both cases the total memory requirement to represent the matrix is $\Theta(n)$." Explain what's right and what's wrong with Professor Prune's argument.

Problem 3. A ternary tree is like a binary tree, except that each node p has three children: left(p), middle(p), and right(p).

1. Give a formula for the number of nodes in a perfect ternary tree of height h.

2. One way to represent a ternary tree is to encode it as a binary tree as shown in Fig. 4.10 on page 110. If T is a perfect ternary tree with n nodes and B is its binary-tree encoding, then how deep is the deepest node of B?

3. One way to represent a complete ternary tree of n nodes is as a table $T[0..n-1]$, similar to the implicit representation of a binary tree described on pages 110 and 111. If i is the index of some node $T[i]$, then give formulas for calculating left(i), middle(i), and right(i).

Problem 4. Suppose a dictionary has n elements whose keys are $K_1, K_2, \ldots, K_n$. Suppose that for each *LookUp* operation, the argument is K_i with probability $1/2^i$. Then the optimal ordering for the keys is $K_1, K_2, \ldots, K_n$. Let $C_{\text{opt}}(n)$ be the expected number of comparisons for a successful search if the keys are stored in that order.

1. When a *LookUp* occurs, what is the probability that the search is unsuccessful?

2. What is $C_{\text{opt}}(4)$?

3. For $n \geq 3$, let $C(n)$ be the expected number of comparisons for a successful search if the keys are stored in the order $K_2, K_3, K_1, K_4, \ldots, K_{n-1}, K_n$. What is $C(n) - C_{\text{opt}}(n)$?

4. It can be proved that $C_{\text{opt}}(n) < 2$ for every n. Use this to give a lower bound on $C(n)/C_{\text{opt}}(n)$.

Problem 4.

1. Write the KMPSkipArray matrix for the string “ABABA”.

2. Write the BMSkipArray matrix for the string ”ABABA”.

Problem 5. Consider a target string that contains 200 occurrences of the character A, followed by the characters ABABA. Suppose we are searching for ABABA.

1. How many times will the Knuth-Morris-Pratt algorithm examine characters in the target?

2. How many times will the Boyer-Moore algorithm examine characters in the target?

Problem 5. Professor Prune says, "Another way to do range searching is to use a binary search tree that is threaded. This way, if want to find all of the nodes in some range, then you only need to search for the *first* node in the range. Once you have found this node, you can follow the threads to find all of the other nodes in the range."

1. Which nodes will Professor Prune's algorithm visit in the tree shown in Figure 9.10 of the book?

2. Is Professor Prune's algorithm any more efficient than ordinary range search? Explain your answer.