**Name: ___________________________________**

**ID Number: ______________________________**

# CSC 225 Midterm Exam

**Oct. 26, 1998**

**Instructions:**

1. Put your name on every page of the exam.
2. No calculators or other aids. Closed book.
3. Read through the entire exam before beginning. You should have 6 pages including this header page.

| **Question** | **Value** | **Mark** |
|---|---|---|
| 1 | 20 | |
| 2 | 20 | |
| 3 | 30 | |
| 4 | 30 | |
| **Total** | **100** | |

Recall that you need at least 40% (40/100) in order to write the final exam in this course.

1. [20] Solve the following recurrence using repeated substitution.
   $T(n) = n + 2\,T(n/2)$, $T(2) = 3$.
   You may assume that $n = 2^k$ for some integer $k \geq 1$.

2. [20] Prove by induction that your solution to question #1 is correct. Or for part marks [10], apply induction to the point where you realize that your solution to #1 is incorrect, and explain what goes wrong.
The recurrence from Question #1:
$T(n) = n + 2\,T(n/2)$, $T(2) = 3$.
You may assume that $n = 2^k$ for some integer $k \geq 1$.

3.(a) [10] State precisely the definition of $O$.

(b) [10] Use your definition from (a) to prove that a polynomial of the form $p(n) = a_0 + a_1 n + a_2 n^2$ is in $O(n^2)$ where $a_i > 0$ for $i = 0, 1, 2$.

(c) [10] Prove that $p(n) = a_0 + a_1 n + a_2 n^2$ is not in $O(n)$ where $a_i > 0$ for $i = 0, 1, 2$.

4. [30] Write detailed pseudocode (almost C code but without worrying about syntax) for **middle_max(n, start, previous_ptr, max_ptr)**. The requirements are the same as assignment #1, and are included on the next page in case you need to reread them. Include lots of comments.

Middle_max takes as input:

**n** - the number of elements on the list L.

**start**- a pointer to the beginning of the linked list L.

Middle_max returns as output:

**previous_ptr**- a pointer to the element before the max in the list or NULL if the maximum element is at the front of the list.

**max_ptr**- a pointer to the maximum element in the list.

The maximum is found by:

1. Divide the problem into two subproblems L1 and L2 where L1 is a list containing only the first floor(n/2) elements of L, and L2 is a list containing the last ceiling(n/2) elements of L.
2. Find M1 the maximum element in L1 and find M2 the maximum element in L2 recursively.
3. Return the larger of M1 and M2.

Create the new L1 and L2 by changing the values of pointers in L, and NOT by allocating any new space for linked list items. At the end, the list L must be restored to its original state.