ECE368 Exam 1
Spring 2010

Wednesday, February 24, 2010
6:30-7:30pm
HORT 117

READ THIS BEFORE YOU BEGIN

This is a *closed book* exam. Electronic devices are not allowed. Since time allotted for this exam is *exactly* 60 minutes, you should work as quickly and efficiently as possible. *Note the allotment of points and do not spend too much time on any problem*.

Always show as much of your work as practical–partial credit is largely a function of the *clarity and quality* of the work shown. Be *concise*. It is fine to use the blank page opposite each question for your work.

This exam consists of 9 pages (including this cover sheet and a grading summary sheet at the end); please check to make sure that *all* of these pages are present *before* you begin. Credit will not be awarded for pages that are missing – it is *your responsibility* to make sure that you have a complete copy of the exam.

IMPORTANT: Write your login at the TOP of EACH page. Also, be sure to *read and sign* the *Academic Honesty Statement* that follows:

"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exercise and will be subject to possible disciplinary action."

Printed Name:

login:

Signature:

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO ...

1. (25 points) $T(n)$, where $n \geq 1$, is defined recursively as

$$T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 3 \cdot \sum_{i=1}^{n-1} T(i) & \text{otherwise.} \end{cases}$$

a. (10 points) Write a recursive algorithm based on the above definition to compute $T(n)$. (Do not attempt to derive a closed-form formula for $T(n)$.)

b. (5 points) Draw a tree to represent the computation of $T(4)$.

c. (5 points) What is the space complexity of your recursive algorithm? Justify your answer.

d. (5 points) What is the time complexity of your recursive algorithm? Justify your answer.

2. (25 points) Show how to implement a deque (double-ended queue) *DQ* using two queues with the following primitives of the queue abstract data type: `EMPTY`(*Q*), `FRONT`(*Q*), `REAR`(*Q*), `ENQUEUE`(*Q*, *element*), and `DEQUEUE`(*Q*)? Implement only the following deque primitives: `EMPTY`(*DQ*), `REMOVE_LEFT`(*DQ*), `REMOVE_RIGHT`(*DQ*), `INSERT_LEFT`(*DQ*, *element*), and `INSERT_RIGHT`(*DQ*, *element*). Assuming $O(1)$ time complexity for all queue primitives, what is the run-time complexity of each of the deque operations.

`EMPTY`(*DQ*)

`REMOVE_LEFT`(*DQ*)

`REMOVE_RIGHT`(*DQ*)

`INSERT_LEFT`(*DQ*, *element*)

`INSERT_RIGHT`(*DQ*, *element*)

3. (20 points)
a. (5 points) Let n be the problem size. True or False: $\log_2 2^n = O(\log_2 2^{n/2})$? Justify your answer.

b. (5 points) Let n be the problem size. True or False: $2^n = O(2^{n/2})$? Justify your answer.

(c) (10 points) Consider the following code fragment:

```
1.      for i ← 1 to n
2.          for j ← i to n
3.              for k ← i to j
```

Write down the respective numbers of times instructions 1, 2, and 3 are executed in terms of i, j, k, and n. Do not attempt to evaluate/expand your expressions.

4. (10 points) Consider the following Shell Sort algorithm for the sorting of an array $r[0..n-1]$ of n integers:

```
for each k in {3,2,1} {
    for j ← k to n−1 {
        temp_r ← r[j]
        i ← j
        while i ≥ k and r[i−k] > temp_r {
            r[i] ← r[i−k]
            i ← i−k
        }
        r[i] ← temp_r
    }
}
```

Consider the array $r = [3,7,9,0,5,1,6,8,4,2,0,6,1,5,7,3,4,9,8,2]$. Show the array r when $k = 2$ and $j = 10$ (right before j becomes 11).

5. (20 points) Consider a binary search tree T implemented with the following data structure:

```
typedef struct _Node {
   int key;
   struct _Node *left;
   struct _Node *right;
} Node;
```

a. (5 points) How many NULL pointers are there in the tree if T has $n \geq 1$ keys (integers) altogether? Justify your answer.

b. (5 points) Consider the following algorithm for constructing a binary search tree from a sorted array `A[0..n]`:

```
Node *Construct_BST(A[], lb, ub) {
     if (lb > ub)
        return NULL;
     mid = (lb+ub)/2;
     node = malloc(sizeof(Node));
     node->info = A[mid];
     node->left = Construct_BST(A[], lb, mid-1);
     node->right = Construct_BST(A[], mid+1, ub);
     return node;
}
```

Draw the binary tree constructed by calling `Construct_BST(A[], 0, 11)` on `A[0..11] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120}`.

c. (5 points) What is the time complexity of the routine `Construct_BST(A[], lb, ub)` when we call `Construct_BST(A[], 0, n-1)` on `A[0..n-1]`? Express the time complexity in terms of *n*. Justify your answer.

d. (5 points) What is the space complexity of the routine `Construct_BST(A[], lb, ub)` when we call `Construct_BST(A[], 0, n-1)` on `A[0..n-1]`? Express the space complexity in terms of *n*. Do not consider the space for the array `A[0..n-1]` and the constructed binary search tree in your expression for the space complexity. Justify your answer.

Question	Max	Score
1	25	
2	25	
3	20	
4	10	
5	20	
Total	100	