ECE368 Exam 1
Spring 2010

Friday, March 26, 2010
9:30-10:20am

READ THIS BEFORE YOU BEGIN

This is a *closed book* exam. Electronic devices are not allowed. Since time allotted for this exam is *exactly* 50 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 8 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) Show how to implement a deque (double-ended queue) *DQ* using two stacks with the following primitives of the stack abstract data type: `EMPTY`(*S*), `STACK_TOP`(*S*), `PUSH`(*S*, *element*), and `POP`(*S*)? 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 stack 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*)

2. (25 points)
a. (5 points) Let n be the problem size. True or False: $(2n)! = O(n!)$? Justify your answer.

b. (5 points) Let n be the problem size. True or False: $\log_2(2n)! = O(\log_2 n!)$? Justify your answer.

(c) (15 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.

3. (20 points) Consider the following Shell Sort algorithm for the sorting of an array $r[0..n-1]$ of n integers:

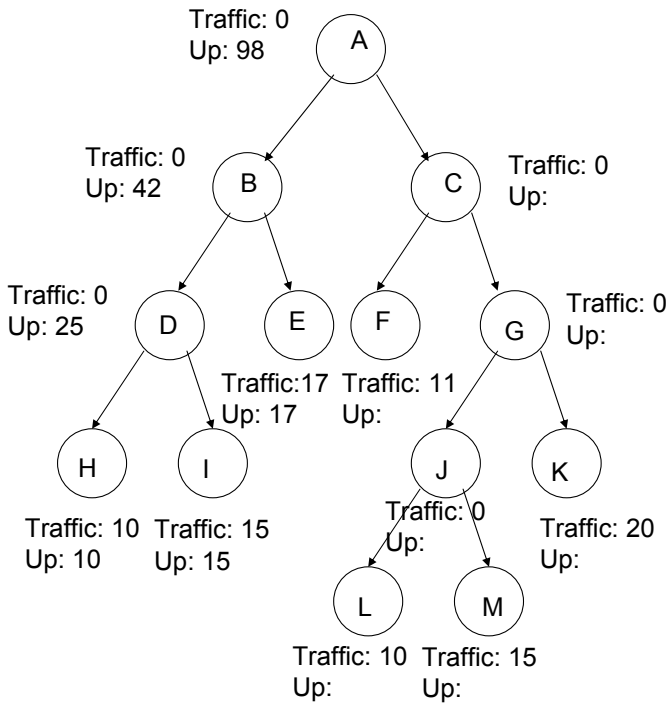
```
for each k in {7,3,1} in decreasing order {
    for j ← k to n−1 {
        temp_r ← r[j]
        i ← j
        while i ≥ k and r[i−k] > temp_r {
            r[i] ← r[i−k]
            i ← i−k
        }
        r[i] ← temp_r
    }
}
```

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 $k = 3$ and $j = 10$ (right before j becomes 11).

4. (30 points) You have been asked to design a computer network that is based on a strictly binary tree topology. You decide to capture the network topology with the following tree data structure:

```
typedef struct _Node {
  char Label;
  int Traffic, Up, Down;
  struct _Node *left;
  struct _Node *right;
} Node;
```

a. (10 points) Suppose you have a tree-based computer network that looks as follows:



In this network, the leaf nodes (E, F, H, I, K, L, and M) are the computers and the internal nodes (A, B, C, D, G, and J) are the routers. The computers generate data traffic but not the routers, which simply transfer data. Therefore, the `Traffic` stored in a leaf node is the amount of data traffic generated by a computer and the `Traffic` stored in an internal node is 0.

The traffic going upstream of a node (computer or router) is stored in the `Up` field of that node. In the worst case, the traffic would have to travel all the way to the root node (before traveling downstream to the other subtree). With that assumption, *the `Up` field of each node is the overall traffic generated by the descendant computers of that node*. For consistency, the `Up` field of the root node is the overall traffic of the entire tree even though it has no upstream link.

The left subtree in the preceding figure has the `Up` fields filled in properly. Fill in the `Up` fields in the right subtree of the preceding figure.

b. (10 points) Modify one or more of the following tree traversal algorithms to compute the `Up` fields, assuming that you are given the `Traffic` fields. (Hint: Determine the relationship between the `Up` fields of a parent node and its child nodes. That relationship will reveal the correct traversal order.)

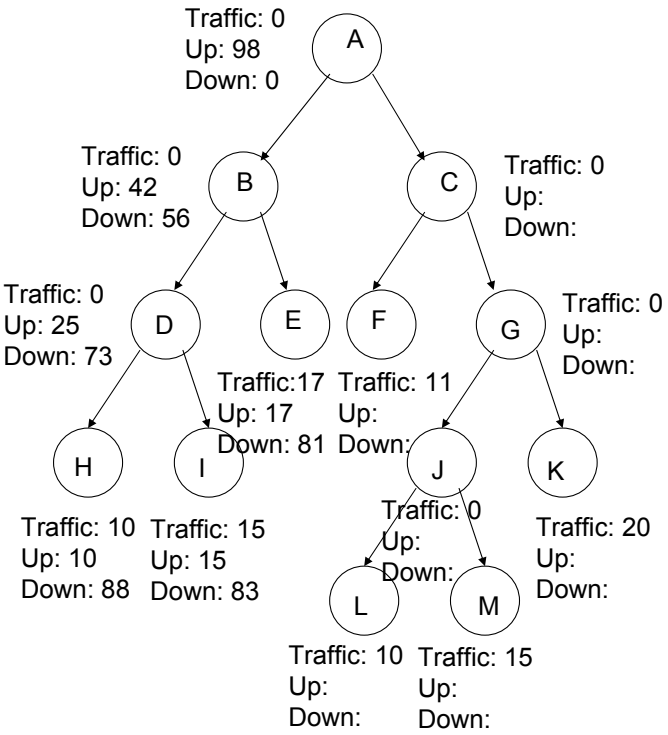
```
Preorder_Traversal(tree_node):
  if (tree_node != NULL) {
    print tree_node
    Preorder_Traversal(left(tree_node))
    Preorder_Traversal(right(tree_node))
  }

Inorder_Traversal(tree_node):
  if (tree_node != NULL) {
    Inorder_Traversal(left(tree_node))
    print tree_node
    Inorder_Traversal(right(tree_node))
  }

Postorder_Traversal(tree_node):
  if (tree_node != NULL) {
    Postorder_Traversal(left(tree_node))
    Postorder_Traversal(right(tree_node))
    print tree_node
  }
```

c. (10 points) The traffic going downtream to a node (computer or router) is stored in the `Down` field of that node. In the worst case, *the traffic flowing downstream to a node is the total traffic from all the non-descendant computers*. The field `Down` field at node D, for example, records the total traffic from non-descendant computers E, F, L, M, and K.

With that assumption, the left subtree in the following figure has the `Down` fields filled in properly. Fill in the `Down` fields in the right subtree of the following figure. (Hint: Determine the relationship between the total traffic in the tree and the `Up` and `Down` fields of a node.)



Which of the three tree traversal algorithms in (b) can be easily modified to compute the `Down` fields? List all of them, but do not write down the algorithm(s). Assume that you already have the `Up` fields properly computed and stored in the tree.

Question	Max	Score
1	25	
2	25	
3	20	
4	30	
Total	100	