Computer Science Foundation Exam

May 7, 2010

Computer Science

Section 1B

Name: SOLUTION

	Max Pts	Type	Passing Threshold	Student Score
Q1	10	ANL	7	
Q2	10	DSN	7	
Q3	10	DSN	7	
Q4	10	ALG	7	
Q5	10	ALG	7	
Total	50		35	

You must do all 5 problems in this section of the exam.

Partial credit cannot be given unless all work is shown and is readable. Be complete, yet concise, and above all be neat. Do your rough work on the last page.

1) (10 points) **Order Notation** Assume that the operations below are implemented as efficiently as possible. Using Big-O notation, indicate the time complexity in terms of the appropriate variables for each of the following operations:

a) Inserting an item into the front of a linked list with n items. **O(1)________**

b) Deleting one node from an AVL tree with n nodes. **O(logn)___**

c) Finding the second smallest integer in an array of n unsorted integers. **O(n)_______**

d) Finding the third largest integer in an array of n sorted integers. **O(1)_______**

e) **Average case** of sorting n integers using Quick Sort **O(nlogn)__**

f) **Best case** for finding the k^{th} smallest integer out of an unsorted array of n integers. ($k \le n$) **O(n)_____**

g) Inserting 5 items into a binary search tree that already has n items. **O(n)______**

h) Printing out all of the permutations of an n letter word with all different letters. (Assume that printing one permutation takes constant time.) **O(n!), or O(n*n!), or O((n+1)!)**

i) Inserting 10 items into an initially **empty** binary heap. **O(1)**

j) Inserting 10 items into a binary heap with n elements. **O(lg n)**

Grading: 1 pt per answer, no partial

2) (10 points) **Linked Lists** Write a function that takes in a pointer to the front of a linked list and determines if the values in that list are stored in **strictly ascending** order. If this is the case, the function should return 1, otherwise it should return 0. In particular, a list is in strictly ascending order if each subsequent value in the list is **greater than** the previous value. The list 2, 5, 6, 7, 10 is in strictly ascending order while the list 2, 4, 4, 7 is not (since 4 isn't greater than 4). Fill in the function below to carry out this task.

```
struct listnode {
     int data;
     struct listnode* next;
};

int ascendingOrder(struct listnode* front)
{

  struct listnode* cur = front;

  if(cur == NULL) return 1;

  while(cur->next != NULL){
    if(cur->data >= cur->next->data){
      return 0;
    }
    cur = cur->next;
  }

  return 1;




// Grading: 3 pts for 0 or 1 node answer
//          4 pts for returning 0 as soon as a discrepancy
//          is found, 3 pts for returning 1 at the end
// Take off 2 pts for a NULL ptr error.









}
```

3) (10 points) **Binary Trees** Write a function that operates on a binary tree. In particular, the function will take in a pointer to the root of the tree and a non-negative integer k. It should return the number of nodes in the tree that are at depth k from the root. (The root is considered to be at depth 0 from the root. The root's children are considered to be at depth 1, grandchildren at depth 2, etc.) Make sure to deal with NULL pointers appropriately.

```
struct treenode
{
     int data;
     struct treenode* left;
     struct treenode* right;
}

// Pre-condition: k >= 0
int numNodesHeightK(struct treenode* root, int k)
{

  if(root == NULL) return 0; // 2 pts
  if(k == 0) return 1; // 2 pts

  return numNodesHeightK(root->left, k-1) +
         numNodesHeightK(root->right, k-1);

  // 1 pt return, 1 pt for each call, 1 pt for each left,
  // right, 1 pt total for k-1 in both calls.

}
```

4) (10 points) **Binary Trees** Examine the function below that makes use of the tree node struct from question 3. Let root be the pointer to the root of the tree shown below. What is the output of the function call *mystery(root)*?

```
void mystery(struct treenode* root) {

    if (root == NULL) return;

    mystery(root->right);
    printf("%d ", root->data%25);
    mystery(root->left);
}
```

```
              35
           /      \
        23          17
       /  \        /
     89   135     56
                 /  \
               44   89
                      \
                       74
                      /
                    287
```

_17__ , _24__ , __12_ , _14__ , _6___ , __19_ , __10_ , _10__ , _23__ , _14__

Grading: 1 pt per blank

5) (10 points) **Recursion** Consider the following recursive function:

```
// Pre-condition: y is non-negative.
int mysterious(int x, int y) {

    if (y == 0) return x;
    return 2*mysterious(x, y-1);
}
```

a) What is the return value of `mysterious(3, 2)`?

12 (3 pts, give 1 pt for either 6 or 24, 0 points otherwise)

b) Mathematically, what function (assuming there are no overflow errors and that y is a non-negative integer) in terms of x and y is mysterious computing? Write out your answer similar to the following: $f(x,y) = 2x + y$. **Justify your answer.**

$f(x,y) = x2^y$ (1 pt for x, 2 pts for 2^y, 1 pt for multiplying them, 3 pts for the explanation)