Computer Science Foundation Exam

December 18, 2009

Computer Science

Section 1B

SOLUTION

PID: _____

	Max Pts	Type	Passing Threshold	Student Score
Q1	11	ANL	8	
Q2	10	DSN	7	
Q3	10	DSN	7	
Q4	9	ALG	6	
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) (11 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) Popping off one element of a stack with *n* items. **O(1)**

b) Inserting three items into an AVL tree of *n* nodes. **O(lg n)**

c) Finding the maximum value in an unsorted array of *n* items. **O(n)**

d) Finding the maximum value in a sorted array of *n* items. **O(1)**

e) Sorting *n* integers using Insertion Sort **(*best case*)** **O(n)**

f) Sorting *n* integers using Insertion Sort **(*worst case*)** **$O(n^2)$**

g) Sorting *n* integers using Quick Sort **(*worst case*)** **$O(n^2)$**

h) Deleting the fifth node (if it exists) from a linked list of *n* items. **O(1)**

i) Multiplying a *n* digit number by a *m* digit number using the standard grade school algorithm. **O(nm)**

j) Determining the height of a binary tree of *n* elements. **O(n)**
(Note: No extra information is stored in a node except for the data in the node and pointers to the left and right children.)

k) The number of moves necessary to solve the Towers of Hanoi with n disks. **$O(2^n)$**

Grading: 1 pt each, all or nothing, if they include leading consts, 1 pt off total, for all instances

2) (10 points) **Linked Lists** Write a function that deletes every other node in the linked list pointed to by the input parameter *head*. (In particular, the second, fourth, sixth, etc. nodes are deleted.)

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

void delEveryOther(struct listnode* head)
{
    // 3 pts
    if (head == NULL || head->next == NULL)
        return;

    // 1 pt
    struct listnode* tmp = head->next;

    // 2 pts
    head->next = tmp->next;

    //  2 pts
    free(tmp);

    // 2 pts
    delEveryOther(head->next);

}
```

Note: Alternate solutions exist, such as an iterative solution. To map the grading criteria from this solution to another, do as follows:

1) Works for lists of size 0 or 1: 3 pts
2) Sets up tmp ptr for iteration: 1 pt
3) Patching around the second node: 2 pts
4) Freeing the second node: 2 pts
5) Some mechanism for repetition: 2 pts

3) (10 points) **Binary Trees** Write a function that operates on a binary tree. (Note: the tree may NOT be a binary search tree.) Your function should return the number of values in the tree less than the input parameter *n*. Make use of the tree node struct and function header below.

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

int numLessThan(struct treenode* root, int n)
{


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

    int sum = 0;

    if (root->data < n)  // 1 pt
        sum++;           // 1 pt

    return sum + numLessThan(root->left, n) +
                 numLessThan(root->right, n);

    // return = 1 pt, root node maybe = 1 pt, each
    // recursive call is worth 2 pts.

}
```

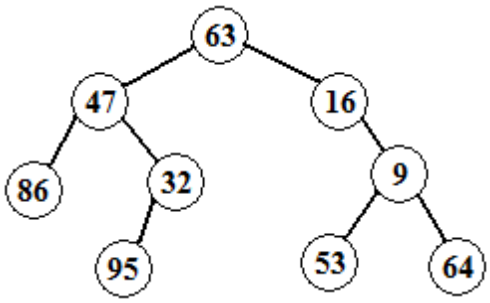
4) (9 points) **Binary Trees** Examine the function below that makes use of the tree node struct from question 3. Let root be the pointer to the root of the tree shown below. What is the output of the function call *mystery(root)*?

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

    if (root == NULL) return;

    mystery(root->right);
    mystery(root->left);

    if (root->data%2 == 0)
        root->data /= 2;
    else {
        int sum = 0;
        if (root->left != NULL) sum += root->left->data;
        if (root->right != NULL) sum += root->right->data;
        root->data += sum;
    }
    printf("%d ", root->data);
}
```



32 , 53 , 94 , 8, 95, 16 , 43 , 106 , 177 (Grading: 1 pt per slot no exceptions)

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

```
void mysterious(int x) {

    if (x == 0) return;

    printf("%d ", x);
    mysterious(x-1);
    mysterious(x-1);
}
```

a) What would be printed by the call to `mysterious(4)`?

4 3 2 1 1 2 1 1 3 2 1 1 2 1 1

Grading: 1 pt for having 15 values, 2 pts, for having the correct frequencies of each, 2 pts for the right order of the values.

b) How many digits will be printed by the call to `mysterious(9)`?

#digits = same as Towers of Hanoi = $2^9 - 1 = 511$.

Grading: 1 pt for recognizing similarity to Hanoi, 3 pts for general formula, 1 pt for correctly plugging in 9.