Computer Science Foundation Exam

May 8, 2009

Computer Science

Section 1B

Name: Grading Criteria

PID: ____________________

	Max Pts	Type	Passing Threshold	Student Score
Q1	10	ANL	7	
Q2	10	DSN	7	
Q3	10	DSN	7	
Q4	10	ANL	7	
Q5	10	ANL	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** Using Big-O notation, indicate the time complexity in terms of the appropriate variables for each of the following operations:

a) Merging two sorted arrays of *n* elements each into a single sorted array ____________

b) Adding *n* elements to an initially empty queue ____________

c) Summing all the numbers in *n* arrays each containing *m* integers ____________

d) Sorting *n* integers using QuickSort (*worst case*) ____________

e) Sorting *n* integers using QuickSort (*best case*) ____________

f) Inserting another integer into an AVL tree containing *n* integers (*best case*) ____________

g) Inserting another integer into a binary search tree containing *n* integers that does not enforce structure properties (*best case*) ____________

h) Inserting another integer into an AVL tree containing *n* integers (*worst case*) ____________

i) Inserting another integer into a binary search tree containing *n* integers that does not enforce structure properties (*worst case*) ____________

j) Playing a complete game of Towers of Hanoi with *n* discs (*best case*) ____________

Solution:

a) O(n)
b) O(n)
c) O(mn)
d) O(n^2)
e) O(n lg n)
f) O(lg n)
g) O(1)
h) O(lg n)
i) O(n)
j) O(2^n)

Grading Criteria:

1 point each, all or nothing

2) (10 points) **Linked Lists** Write a function that operates on a linked list of integers. Your function should delete any node that contains an odd integer and return a pointer to the front of the resulting list. Make use of the list node struct and function header below.

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

struct listnode*  del_odd_nodes(struct listnode* head)
{
```

Solution:

```
     struct listnode* temp;
     while(head != NULL && head->data % 2 == 1){
          temp = head;
          head = head->next;
          free(temp);
     }
     if(head != NULL)
          del_odd_nodes(head->next);

     return head;
```

Grading Criteria:

There are many possible solutions to this question, some involving recursion, some not. Be reasonable when grading this question.
2 points for freeing the memory
4 points for correctly removing a node in a way that is otherwise correct
4 points for deleting the correct nodes and only the correct nodes

```
}
```

3) (10 points) **Binary Trees** Write a function that operates on a binary search tree. Your function should delete the node storing the minimum value in the tree and return a pointer to the root of the resulting tree. Note: If the initial pointer passed in is NULL, simply return NULL. Make use of the tree node struct and function header below.

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

struct treenode*  delete_min(struct treenode* root)
{
```

Solution:

```
     struct treenode* temp;
     if(root == NULL)
          return NULL;
     if(root->left != NULL){
          root->left = delete_min(root->left);
          return root;
     }
     temp = root->right;
     free(root);
     return temp;
```

Grading Criteria:

There are many possible solutions to this question. Be reasonable when grading this question.

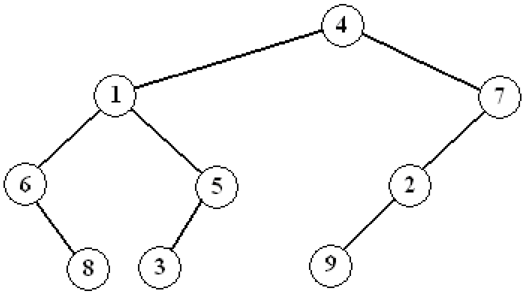
2 points for handling a null root
3 points for correctly locating the node to delete
1 point for not trying to delete incorrect nodes
3 points for removing the target node from the tree
1 point for freeing the memory of the target node

```
}
```

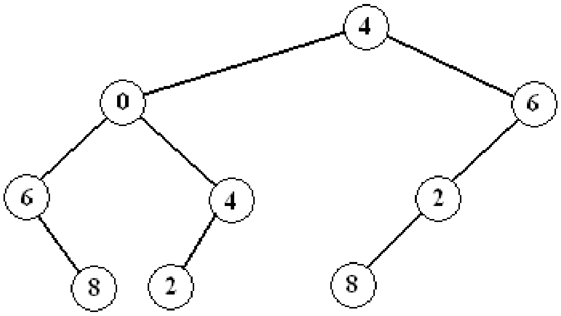
4) (10 points) **Binary Trees** Examine the function below that makes use of the tree node struct from question 3.

```
int mystery(struct treenode* root) {
      int rval;
      if(root == NULL)
            return 0;
      rval = mystery(root->left) + mystery(root->right);
      if(root->data % 2 == 1){
            root->data -= 1;
            rval++;
      }
      return rval;
}
```

a) Briefly explain what the function does and what its return value means.
b) Show the state of the tree below after mystery is called on its root and indicate the value returned by the function.



Solution:
a) The function subtracts 1 from all nodes containing odd values and returns the number of nodes that are altered by the function.
b)



Grading Criteria:
a) (6 points total)
4 points for determining the purpose of the function
2 points for determining the meaning of the return value

b) (4 points total)
2 points for correctly altering the nodes that should be altered
2 points for not altering other nodes

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

```
void mysterious(int x) {
     if(x == 0)
          return;
     printf("%d %d\n", x % 2, x);
     mysterious(x/2);
}
```

a) What would be printed by the call to `mysterious(6)`?
b) What would be printed by the call to `mysterious(42)`?

Solution:
a)

```
0 6
1 3
1 1
```

b)

```
0 42
1 21
0 10
1 5
0 2
1 1
```

Grading Criteria:
5 points per part
2 points for the left column
2 points for the right column
1 point for the answer being otherwise correct