Computer Science Foundation Exam

August 14, 2009

Computer Science

Section 1B

Name:____

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** 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 every element off a stack containing *n* elements ____________
b) Adding *m* elements to a queue that already contains *n* elements ____________
c) Computing the arithmetic mean (average) of an array containing *n* integers ____________
d) Determining the median in a sorted array containing *n* integers ____________
e) Sorting *n* integers using Mergesort (*best case*) ____________
f) Inserting *n* integers into an initially empty AVL tree (*best case*) ____________
g) Inserting *n* integers into an initially empty AVL tree (*worst case*) ____________
h) Inserting *n* integers into an initially empty binary search tree that does not enforce structure properties (*best case*) ____________
i) Inserting *n* integers into an initially empty binary search tree that does not enforce structure properties (*worst case*) ____________
j) Deleting every other node (i.e. node 2, 4, 6, 8, …) from a linked list containing *n* elements ____________

Solution:
a) $O(n)$
b) $O(m)$
c) $O(n)$
d) $O(1)$
e) $O(n \log n)$
f) $O(n \log n)$
g) $O(n \log n)$
h) $O(n \log n)$
i) $O(n^2)$
j) $O(n)$

Grading Criteria:
1 point each

---

2) (10 points) **Linked Lists** Write a function that operates on a linked list of integers. Your function should insert a new node containing the value 2 after every node that contains the value 4. Make use of the list node struct and function header below.

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

void list_42(struct listnode* head)
{
```

Solution:

```
	struct node* crnt = head;
	struct node* temp;

	while(crnt != NULL)
	{
		if(crnt->data == 4)
		{
			temp = malloc(sizeof(struct node));
			temp->data = 2;
			temp->next = crnt->next;
			crnt->next = temp;
		}
		crnt = crnt->next;
	}
```

Grading Criteria:

There are many possible solutions to this question, some involving recursion, some not. Be reasonable when grading this question.

2 points for traversing the list correctly
1 point for determining where to stop traversing
2 points for determining where to insert nodes
1 point for allocating memory for the new node
1 point for setting the data of the new node
3 points for linking the new node into the list correctly

3) (10 points) **Binary Trees** Write a function that operates on a binary tree. Your function should delete all leaf nodes from the original tree and return a pointer to the root of the adjusted tree. Make use of the tree node struct and function header below.

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

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

Solution:

```
	if(root == NULL)
		return NULL;
	if(root->left == NULL && root->right == NULL){
		free(root);
		return NULL;
	}
	root->left = delete_leaves(root->left);
	root->right = delete_leaves(root->right);
	return root;
```

Grading Criteria:

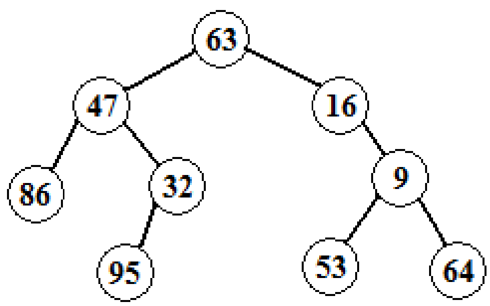
There are many possible solutions to this question. Be reasonable when grading this question.

1 point for handling a null root
2 points for detecting leaves
1 point for freeing the leaf
2 point for removing the leaf from the tree
3 points for correct recursive calls
1 point for the correct return value in non-leaf cases

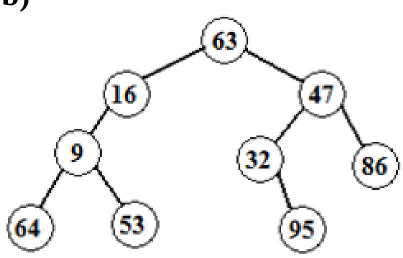
4) (10 points) **Binary Trees** Examine the function below that makes use of the tree node struct from question 3.

```
int mystery(struct treenode* root) {
      struct treenode* temp;

      if(root == NULL)
            return 0;

      temp = root->left;
      root->left = root->right;
      root->right = temp;

      return 1 + mystery(root->left) + mystery(root->right);
}
```

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 flips the tree left-to-right (i.e. mirrors the tree). The function returns the number of nodes in the tree.
b)



Return value: 9

Grading Criteria:
4 points for determining what the function does
2 points for determining the meaning of the return value
3 points for producing the correct tree in part b
1 point for producing the correct return value in part b

---

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

```
void mysterious(int x) {
     int i;
     if(x < 2)
          return;

     for(i = 2; i <= x; i++){
          if(x % i == 0){
               printf("%d ", i);
               mysterious(x / i);
               return;
          }
     }
}
```

a) What would be printed by the call to `mysterious(24)`?
b) What would be printed by the call to `mysterious(90)`?

Solution:
a) 2 2 2 3
b) 2 3 3 5

Grading Criteria:
5 points per part:
Producing the correct numbers – 3 points
Producing the numbers in the correct order – 2 points
