COP 3502 – Spring 2012
Exam #2 (Solutions)
3/30/12

Exam 2 Comments:
Coming

Statistics:

# of Exams Taken:	**195**
Average Grade:	**85 (which is a HIGH average)**
# of Perfect Scores:	**16**
# of Grades >= 100:	**44**
# of Grades >= 90:	**94**
# of Grades >= 80:	**131**

1) (15 pts) **Recurrence Relations.** Solve the following recurrence relation:
****Note: in order to get full credit, **you MUST show your work**, as shown in class and on the practice sheet on the course website. Also, you must "iterate" at least two times.*

$T(n) = T(n/4) + 12$ | Substituting Equations

$T(1) = 7$

$T(n) = T(n/16) + 12 + 12$
$T(n) = T(n/64) + 12 + 12 + 12$
$T(n) = T(n/256) + 12 + 12 + 12 + 12$
$T(n) = T(n/1024) + 12 + 12 + 12 + 12 + 12$

$\underline{n \rightarrow n/4}$

$T(n/4) = T(n/16) + 12$
$T(n/16) = T(n/64) + 12$
$T(n/64) = T(n/256) + 12$
$T(n/256) = T(n/1024) + 12$

So now rewrite these five equations and look for a pattern:

$T(n) = T(n/4^1) + 12*1$ ⟵ 1st step of recursion
$T(n) = T(n/4^2) + 12*2$ ⟵ 2nd step of recursion
$T(n) = T(n/4^3) + 12*3$ ⟵ 3rd step of recursion
$T(n) = T(n/4^4) + 12*4$ ⟵ 4th step of recursion
$T(n) = T(n/4^5) + 12*5$ ⟵ 5th step of recursion

Generalized recurrence relation at the kth step of the recursion:
$T(n) = T(n/4^k) + 12*k$

Let $n = 4^k$. Solving for k, we get $k = \log_4 n$. Plug in.
$T(n) = T(4^k / 4^k) + 12\log_4 n = T(1) + 12 \log_4 n = 7 + 12 \log_4 n$

$T(n) = 7 + 12 \log_4 n$

2) (10 pts) **Stack Applications.** Convert the following infix expression into its equivalent postfix expression using a stack. Additionally, you must show the contents of the stack at the indicated points (1, 2, and 3) in the infix expression.

```
          1               2                   3
( A + ( B * C ) + D ) / ( E + F ) * G - H / I
```

1	2	3
*		
(		
+	(	/
(	/	-

Resulting Postfix expression: A B C * + D + E F + / G * H I / -

3) (8 pts) **Stack Applications.** Evaluate the following postfix expression using a stack. Additionally, you must show the contents of the stack at the indicated points (1, 2, and 3) in the postfix expression.

```
          1          2          3
4 12 6 / 4 8 – 5 6 4 – * – – +
```

1	2	3
	6	
	5	
4	-4	-14
2	2	2
4	4	4

Final Answer: 20

4) (5 points) **Queues 'n Stacks.** Let `q` be a queue and `s` be a stack. The functions `dequeue` and `pop` obey the convention that they return whatever they remove. Assume that `q` and `s` are initially empty and that `i` has been declared as an `int`. What would be **printed** by the following code fragment? (put answer in the box)

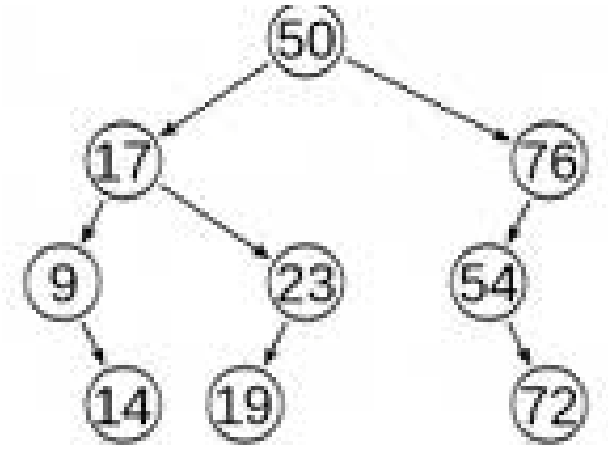
```
push(s, 7);
push(s, 4)
enqueue(q, 3);
enqueue(q, 5);
for(i = 0; i < 4; i++){
     printf("%d  ", pop(s));
     printf("%d  ", dequeue(q));
     push(s, i*5);
     enqueue(q, i+5);
}
```

***Also, **after** you exit the for loop, show the remaining contents of the stack, s, in the boxes below:

15
7

Answer: **4 3 0 5 5 5 10 6**

5) (20 points) **Binary Tree Traversals.** Give the preorder, inorder, postorder, and breadth-first traversals of the binary tree shown below. Do the breadth first as shown in class and on slides.
**Note: this is not a Binary Search Tree. However, that does not affect the problem or solution.*



Preorder: **50 17 9 14 23 19 76 54 72**

Inorder: **9 14 17 19 23 50 54 72 76**

Postorder: **14 9 19 23 17 72 54 76 50**

Breadth-first: **50 17 76 9 23 54 14 19 72**

6) (16 points) **Binary Tree Code.** Write a **recursive** function that operates on a binary tree of integers. Your function should SUM up all of the ODD **numbers** in the tree EXCEPT for the numbers in the leaf nodes, and then should return that value. Your function should make use of the following `struct tree_node` and function prototype:

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

int sum_nonleaf_odd(struct tree_node *p) {

      if (p == NULL) return 0;

      if (p->left == NULL && p->right == NULL)
            return 0;

      int sum = 0;

      if (p->data%2 == 1)
            sum += p->data;

      sum += sum_nonleaf_odd(p->left);

      sum += sum_nonleaf_odd(p->right);

      return sum;









}
```

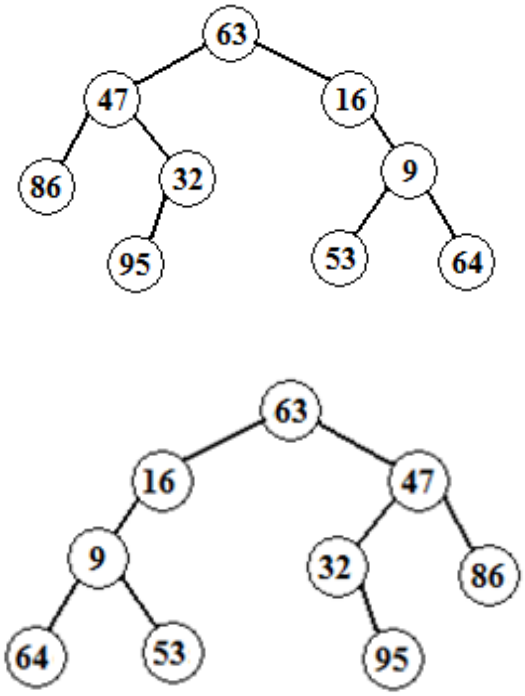
7) (10 points) **Binary Trees.** Examine the function below that makes use of the typical `tree_node` struct from question 6:

```
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.

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

b) Show (**redraw**) the state of the tree below **after** mystery is called on its root, **AND** indicate the value returned by the function.



Return Value: 9

8) (8 points) **Sorting.** Show the contents of the array below being sorted using **<u>Insertion Sort</u>** at the end of each loop iteration. **Reminder*: the loop iterates from i = 1 to i = n-1.*

Index	0	1	2	3	4	5
Initial Array:	**14**	**42**	**38**	**7**	**22**	**47**
	14	42	38	7	22	47
	14	38	42	7	22	47
	7	14	38	42	22	47
	7	14	22	38	42	47
Sorted Array:	**7**	**14**	**22**	**38**	**42**	**47**

9) (8 points) **Sorting.** Show the contents of the array below being sorted using **<u>Selection Sort</u>** at the end of each loop iteration. As shown in class, please run the algorithm by first placing the smallest item into its correct place, and then the next smallest, and so on.

Index	0	1	2	3	4	5
Initial Array:	**21**	**7**	**42**	**14**	**28**	**35**
	7	21	42	14	28	35
	7	14	42	21	28	35
	7	14	21	42	28	35
	7	14	21	28	42	35
Sorted Array:	**7**	**14**	**21**	**28**	**35**	**42**

10) (5 points) **Freebie.** What is the worst movie you've ever seen?

2001: A Space Odyssey. This is the only correct answer, period.