Name(s) ____________________ CSE Login ____________________

Instructions Same partner/group policy as the project applies to this assignment. Answer each question as completely as possible. You are encouraged to typeset your assignment using LaTeX or some typesetting system. Figures may be hand drawn and scanned/included as an image.

Question	Points	Score
1	15	
2	15	
3	5	
4	5	
5	24	
6	4	
7	9	
8	8	
9	8	
10	2	
11	30	
12	0	
Total:	125	

Algorithm Analysis

When asked to design and analyze an algorithm, be sure to provide the following:

1. Complete pseudocode
2. Identify the input and the input size, n
3. Identify the elementary operation
4. Compute how many times the elementary operation is executed with respect to the input size n
5. Provide a Big-O asymptotic characterization for the algorithm's complexity

1. $\boxed{\text{15 points}}$ Let P be an image represented as an $(n \times m)$ 2-dimensional array of pixels. Design and analyze an algorithm that given an image P will rotate it counter clockwise by 90 degrees.

2. $\boxed{\text{15 points}}$ Let R be a set of rectangles in the 2-D plane represented as a pair of points, $((x, y), (x', y'))$ where (x, y) is the lower left point of the rectangle and (x', y') is the upper right point of the rectangle. Design and analyze an algorithm that given a set R of n rectangles finds the pair of rectangles whose intersection has the greatest area.

For the next few questions, consider defining a *time function* with respect to n. That is, if the algorithm is linear, we could write the time that it takes as a function of n:

$$t = f(n) = cn$$

If it is quadratic, we could write it as a quadratic function:

$$t = f(n) = cn^2$$

Both of these instances ignore lower order terms. If we know the time it takes for a particular value of n then we can compute the constant c and consequently predict the time t it takes for other values of n or vice versa.

3. $\boxed{\text{5 points}}$ An algorithm takes 2.5 ms for an input size 100. How long will it take for input size 2000 (assuming that low-order terms are negligible) if the running time is
   (a) linear
   (b) $O(n \log n)$
   (c) quadratic
   (d) cubic
   (e) exponential

4. $\boxed{\text{5 points}}$ An algorithm takes 1 ms for input size 100. How large can an input size be if a problem can be solved in 1 minute (assuming that low-order terms are negligible) if the running time is:
   (a) linear
   (b) $O(n \log n)$
   (c) quadratic
   (d) cubic
   (e) exponential

5. $\boxed{\text{24 points}}$ Prove each of the following statements by applying the definition of Big-O. That is, derive an inequality (show your work) and clearly identify the c, n_0 constants you derive as per the definition of Big-O.

(a) $3n = O(n)$

(b) $500\sqrt{n} = O(n)$

(c) $n^2 + 2n + 1 = O(n^2)$

(d) $2048n + 1234 = O(n^2)$

(e) $n \log(32n) = O(n \log(n)))$

(f) $12n^3 + 50n^2 - 12n - 60 = O(n^3)$

(g) $n2^n = O(3^n)$

(h) $\log(n!) = O(n \log n)$

Recursion

6. $\boxed{\text{4 points}}$ (Weiss 7.17) Give a Big-O characterization for the following recurrences (with initial conditions $T(0) = T(1) = 1$.

   (a) $T(n) = T(n/2) + 1$

   (b) $T(n) = T(n/2) + n$

   (c) $T(n) = T(n/2) + n^2$

   (d) $T(n) = 3T(n/2) + n$

   (e) $T(n) = 3T(n/2) + n^2$

   (f) $T(n) = 4T(n/2) + n$

   (g) $T(n) = 4T(n/2) + n^2$

   (h) $T(n) = 4T(n/2) + n^3$

Trees

7. $\boxed{\text{9 points}}$ Perform a pre-order, in-order, and post-order traversal on the tree in Figure 1 and give the resulting node sequences for each.

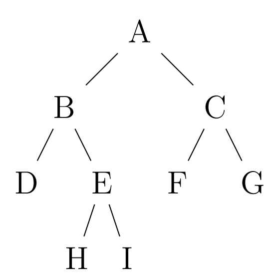

Figure 1: A Tree

8. $\boxed{\text{8 points}}$ Consider a binary search tree T with 5 nodes.
   (a) What is the *maximum* possible depth of T?
   (b) What is the *minimum* possible depth of T?
   (c) What are the minimum number of leaves T can have?
   (d) What are the maximum number of leaves T can have?

9. $\boxed{\text{8 points}}$ Consider a binary search tree T with 15 nodes.
   (a) What is the *maximum* possible depth of T?
   (b) What is the *minimum* possible depth of T?
   (c) What are the minimum number of leaves T can have?
   (d) What are the maximum number of leaves T can have?

10. $\boxed{\text{2 points}}$ Consider a binary search tree T with n nodes.
   (a) What is the *maximum* possible depth of T?
   (b) What is the *minimum* possible depth of T?

11. When dealing with trees, algorithm complexity is usually measured in terms of the number of nodes in the tree while the elementary operation is usually a node traversal. Let T be a tree such that each node u has a `parent`, `rightChild` and `leftChild`.
   (a) $\boxed{\text{15 points}}$ Design and analyze an algorithm (provide pseudocode) that, given a node u in T determines its depth d. Analyze your algorithm.
   (b) $\boxed{\text{15 points}}$ Design and analyze an algorithm (provide pseudocode) that, given an integer d and a tree T with nodes $u_1, \ldots, u_n$ determines the *number* of nodes in T at depth d.

Honors/Bonus

These are required for students in the honor(s) section, they are bonus for those in the main section.

12. $\boxed{\text{0 points}}$ When dealing with trees, algorithm complexity is usually measured in terms of the number of nodes in the tree while the elementary operation is usually a node traversal. Let T be a binary tree such that each node u has a `parent`, `rightChild` and `leftChild` and a *key* element, `key`. The left/right child may be `null` indicating it does not exist.

   Given a binary tree, we wish to determine if it is a binary search tree or not.

(a) (5 points) Gomer thinks we can solve this problem by using a preorder traversal. When each node u is processed, it simply verifies that the left child's key is strictly less than u's key and u's right child's key is strictly greater than u's key. Gomer's solution, however, will not work. Provide a counter example to demonstrate this and explain how it shows Gomer is wrong.

(b) (20 points) Design a *correct* algorithm that, given a root note in a binary tree (with `parent`, `rightChild`, `leftChild` and `key` elements) determines if it is a binary search tree or not. Fully analyze your algorithm.