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. Hand your answers in hardcopy with a copy of this coversheet and all your name(s).

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>134</td>
<td></td>
</tr>
</tbody>
</table>
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. 10 points Let \( P \) be an image represented as an \((n \times m)\) 2-dimensional array of pixels. Design an algorithm that given an image \( P \) will rotate it clockwise by 90 degrees.

2. 10 points Let \( C \) be a set of circles each represented as a triple \((x, y, r)\) where \( x, y \) is its center and \( r \) is its radius. Design and analyze an algorithm that given a set \( C \) of \( n \) circles determines if any of the circles intersect.

3. 10 points Let \( A = [a_1, a_2, \ldots, a_n] \) be a collection of integers. A pair \((i, j)\) is called an inversion if \( i < j \) but \( a_i > a_j \). For example, if \( A = [2, 3, 8, 6, 1] \) then the list of inversions is \((1, 5), (2, 5), (3, 4), (3, 5), (4, 5)\). Design an algorithm (provide good pseudocode) that, given a collection of integers \( A \) outputs a list of its inversions. Provide a complete analysis of your algorithm.

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.

4. 5 points An algorithm takes 1.5 ms for an input size 100. How long will it take for input size 1500 (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. **5 points** An algorithm takes 0.5 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

6. **8 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) \( \log (n!) = O(n \log n) \)

7. **6 points** (Weiss 5.19) Order the following functions by growth rate:

\( n, \sqrt{n}, n^{1.5}, n^2, n \log n, n \log (\log (n)), n \log^2 (n), n \log (n^2), 2^n, 37, n^3, n^2 \log n \)

Recursion

8. **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) \( T(n) = T(n/2) + n \)
Stacks & Queues

9. **10 points** Say that you need to design a Queue ADT, but you do not have access to the usual building blocks (Lists, Arrays, etc.), you only have access to a Stack ADT. Describe, by providing pseudocode how you would implement the enqueue and dequeue operations using two stacks.

10. **10 points** An XML document is a plain text file containing opening and closing tags that semantically mark-up pieces of data. Further, tags can be nested as many levels as necessary. Use an appropriate data structure to design an algorithm (provide good pseudocode) to determine whether or not an XML document is well-balanced: that is all opening tags have a corresponding closing tag and such tags are not illegally interleaved.

Trees

11. **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.

```
        A
       / \  / \  
      B   C D   E  / \  /  
     /   / F   G H   I 
```

Figure 1: A Tree

12. **5 points** (Weiss 19.1) Show the result of inserting the following elements in the following order into an initially empty binary search tree. Then show the result of deleting the root.

    3, 1, 4, 6, 9, 2, 5, 7
13. **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?

14. **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?

15. **4 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 $T$?

16. **8 points** Say that we want to insert integers 1 thru 15 into a binary search tree one by one. What order would we have to insert these elements for it to result in a full and balanced binary search tree?

17. **15 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) 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) 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.

**Honors/Bonus**

These are required for students in the honor(s) section, they are bonus for those in the main section.
18. (Bonus/Honors 10pts) Let $A$ be a zero-index array that is sorted. Now suppose that someone comes along and cyclicly “shifts” the contents of $A$ so that the contents are moved some number of positions, with those at the end being shifted back to the beginning. For example, if the array contained the elements $2, 6, 10, 20, 30, 40$ and it were shifted by 3 positions, the resulting array would be $20, 30, 40, 2, 6, 10$ (in general, you will not be given how many positions the array has been shifted).

Design an efficient algorithm that can compute the median element of $A$. Give good pseudocode and a short explanation. Analyze your algorithm. Full credit will only be given to solutions that are $O(\log n)$.

19. [0 points] (Bonus/Honors 10pts) Recall that a binary search tree is a binary tree with keys at each node such that for each node with key $k$, all keys in the left-sub-tree are less than $k$ and all keys in the right-sub-tree are greater than $k$ (assume all keys are unique). The usual in-order traversal produces a key sequence in non-decreasing order.

Write an algorithm (give good pseudocode and a complete analysis) that inverts a binary search tree so that for each node with key $k$, all keys in the left-sub-tree are greater than $k$ and all nodes in the right-sub-tree are less than $k$. Thus, an in-order traversal of an inverted tree will produce a key ordering in non-increasing order instead. An example of a binary search tree and its inversion is presented in Figure 2. Your algorithm must be efficient and should not create a new tree.