You are to honor the Emory Honor Code and the Math/CS SPCA. This is a closed-book and closed-notes exam. You have 150 minutes to complete this exam.

### Short answers

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>Earned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Long answers

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>Earned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Pseudo-code completion

<table>
<thead>
<tr>
<th></th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>40</td>
</tr>
<tr>
<td>Earned</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Total points
Short Answers

1. The worst-case complexities of SmartSelect and HeapSelect are $O(n \log k)$ and $O(n \log n)$ for selecting the $k$'th maximum key in a list. However, given our implementation, we found that HeapSelect outperformed SmartSelect as $k$ gets closer to $n$. Explain why (2 points).

2. The worst-case complexities of MergeSort and QuickSort are $O(n \log n)$ and $O(n^2)$. However, given our implementation, we found that QuickSort always outperformed MergeSort even for the worst cases. Explain why (2 points).

3. What are the worst-case complexities of the following algorithms? Explain how you derive these complexities (6 points).
   
   (a) Kruskal’s minimum spanning tree algorithm.
   (b) Chu-Liu-Edmonds’ minimum spanning tree algorithm.
   (c) Dijkstra’s shortest-path algorithm.
   (d) Ford-Fulkerson’s maximum flow algorithm.

4. Can Prim’s algorithm be used for finding minimum spanning trees in directed graphs? Explain why (2 points).

5. Which complexity class (P, NP, NP-complete, NP-hard) does the algorithm in homework #3 that finds all minimum spanning trees belong to? Explain why (2 points).

6. What is the worst-case complexity of detecting a cycle in a directed graph? Explain why (2 points).

7. Prove that the distances[$u$] in our Dijkstra’s algorithm is the minimum length between the vertex $u$ and the target vertex $t$ (3 points).

8. Explain why Ford-Fulkerson’s algorithm without backward pushing in directed graphs would not always find the optimum solutions (2 points).

9. (a) Explain why a matching is not maximum if there exists an augmenting path (2 points).
   
   (b) Prove any maximum matching is also a maximal matching in the same graph (3 points).

10. (a) Are problems in NP-hard solvable in polynomial time if $P = NP$? Explain why (2 points).
    
    (b) What are the properties of problems in NP-complete if $P \neq NP$, besides that they are the hardest problems in NP (2 points)?
Long Answers

11. Given the list of [120, 34, 201, 344, 13, 320, 1, 132], show how it can be sorted by RadixSort using the most significant digit (e.g., “1” is the most significant digit of “123”). Describe each sorting step (e.g., which key goes into which bucket when using which digit). Note that MSD RadixSort is slightly different from LSD RadixSort (6 points).

12. Given the input of [5, 3, 4, 7, 6], show how a Red-Black tree balances when each key is added in order. Describe how the color and hierarchy of each node changes (6 points).

13. Prove that the maximum flow is equal to the capacity of a minimum cut in the same graph (6 points).

14. Given the following graph, define (a) the objective function and (b) the necessary constraints for linear programming that finds the minimum cut. Assume that each flow amount is an integer value. Explain what the variables imply in your linear programming (6 points).

![Graph](image1)

15. Given the following bipartite graph, define (a) the objective function and (b) the necessary constraints for linear programming that finds the maximum matching (6 points). Explain what the variables imply in your linear programming (6 points).

![Bipartite Graph](image2)
Pseudo-code Completion

Use only methods in provided interfaces, classes, and Java built-in APIs for all questions. Extra codes that either hurt or do not contribute to the assigned tasks will be penalized.

16. Write a method in pseudo-code that takes two strings, a and b, and finds a longest common subsequence. Your code must show (a) how to populate a dynamic table and (b) how to find a longest common subsequence by searching through the dynamic table (8 points).

```java
/**
 * @param a the first string.
 * @param b the second string.
 * @return a longest common sequence of the specific strings a and b.
 */
public String findLCS(String a, String b) {
    // To be filled.
}
```

17. Write a method in pseudo-code for topological sort using depth-first search. Your code must return the list of integers representing vertices in topological order (8 points).

```java
public interface Graph
{
    Deque<Integer> getAllVerticesWithNoIncomingEdges();
    List<Edge> getIncomingEdges(int vertex);
    List<Edge> getOutgoingEdges(int vertex);
    boolean isEmpty(); int size();
}

public interface Edge
{
    int getSource(); double getWeight();
    int getTarget(); void setWeight(double weight);
}

/**
 * @param graph a graph containing vertices to be sorted.
 * @return the list of integers representing vertices in topological order.
 */
public List<Integer> topologicalSort(Graph graph) {
    // To be filled.
}
```
18. Write two methods, `rotateRight` and `balance`, in pseudo-code for AVL trees. The `rotateRight` method takes a node, rotates it to the right, and resets all necessary heights. The `balance` method takes a node and balances the AVL tree using rotations (12 points).

```java
public interface AVLNode {
    AVLNode getParent(); void setParent(AVLNode node); boolean hasParent();
    AVLNode getLeftChild(); AVLNode getRightChild();
    boolean hasLeftChild(); boolean hasRightChild();
    void setLeftChild(AVLNode node); void setRightChild(AVLNode node);
    void replaceChild(AVLNode oldChild, AVLNode newChild);
    int getBalanceFactor(); int getHeight(); void setHeight(int height);
}

public class AVLTree {
    AVLNode n_root;

    /** Rotates the node to the left and resets all necessary heights. */
    public void rotateLeft(AVLNode node) {
        // Do not fill this. Assume this method exists.
    }

    /** Rotates the node to the right and resets all necessary heights. */
    public void rotateRight(AVLNode node) {
        // To be filled.
    }

    /** Balances the AVL tree after adding or removing the specific node. */
    public void balance(AVLNode node) {
        if (node == null) return;
        int bf = node.getBalanceFactor();

        if (bf == 2) {
            // Do not fill this.
        } else if (bf == -2) {
            // To be filled.
        } else {
            balance(node.getParent());
        }
    }
}
```
19. (a) Describe how Chu-Liu-Edmonds’ algorithm works (not in pseudo-code; 4 points).

(b) Write a method in pseudo-code for Chu-Liu-Edmonds’ algorithm that takes a graph and a list of cycles, and updates the weights related to the edges in the cycles. Use the Graph and Edge interfaces from question #17 (8 points).

```java
/**
* @param graph a graph containing a minimum spanning tree.
* @param cycles the list of cycles where each cycle is represented as a list of edges.
* Updates all weights related to the edges in the cycles.
*/
public void updateEdgeWeights(Graph graph, List<List<Edge>> cycles) {
    // To be filled
}
```