1. True or False. [5 points each]
   (a) T  If an AVL tree has $n$ nodes, then its height is $\Theta(\log n)$.
   (b) T  Dijkstra’s algorithm is a special case of the A* algorithm.

2. Fill in the blanks. [5 points each blank.]
   (a) A graph of 6 vertices has no more than 15 edges. (Exact answer.)
   (b) An acyclic digraph with 20 arcs has at least 7 vertices. (Exact answer.)
   (c) A planar graph with 100 edges has at least 36 vertices. (Exact answer.)
   (d) If a directed graph with 6 vertices has exactly two strong components of the same size, and if it has
       no self-loops, then it can have no more than 21 arcs. (Exact answer.)
   (e) [10 points] For any shortest path problem on a weighted directed graph $G$, $G$ can have no negative
cycle. (Two words.)

3. Let $G$ be a directed graph with $n$ vertices and $m$ arcs. Fill in the blanks, using asymptotic notation. [5
points each blank.]
   (a) [5 points] The time complexity of the Floyd-Warshall algorithm is $\Theta(n^3)$.
   (b) [10 points] The time complexity of Dijkstra’s algorithm is $O(m \log n)$, but there can be no negative
arcs.
   (c) [5 points] The time complexity of Johnson’s algorithm is $O(nm \log n)$.

4. What is the asymptotic time complexity, in terms of $n$, of each of the following code fragments? (10
points each)
   (a) for(int $i = 1$; $i < n$; $i++$)  \[ \Theta(n) \]
       for(int $j = n$; $j > i$; $j=j/2$)

   (b) for(int $i = 1$; $i < n$; $i=i*2$)  \[ \Theta(n \log n) \]
       for(int $j = i$; $j < n$; $j++$)
5. Solve the recurrences. Give asymptotic answers in terms of $n$. Use $O$, $\Omega$, or $\Theta$, whichever is most appropriate. [10 points each.]

(a) $F(n) \leq 4F(n/2) + 1$ \quad $F(n) = O(n^2)$
(b) $H(n) = 2H(n/4) + \sqrt{n}$ \quad $H(n) = \Theta(\sqrt{n} \log n)$
(c) $K(n) = K(n/2) + K(n/4) + n$ \quad $K(n) = \Theta(n)$
(d) $L(n) < L(\sqrt{n}) + 1$ \quad $L(n) = O(\log \log n)$

6. The following C++ code computes a function $F$.

```cpp
int F(int n) // input condition: n >= 0
{
    if(n == 0) return 0;
    else return 4*F(n/4)+3*F(n/2) + n*n;
}
```

(a) [10 points] What is the asymptotic complexity of $F(n)$?

- The recurrence is $F(n) = 4F(n/4) + 3F(n/2) + n^2$
- The solution, by the generalized master theorem, is $F(n) = \Theta(n^2 \log n)$.

(b) [10 points] What is the asymptotic time complexity of the recursive computation of $F(n)$ given by the code above?

- Let $T(n)$ be the time complexity of the code. The recurrence is $T(n) = 4T(n/4) + 3T(n/2) + 1$.
- The solution, by the generalized master theorem, is $T(n) = \Theta(n^2)$

(c) [10 points] What is the asymptotic time complexity of the dynamic programming computation of $F(n)$?

- Let $D(n)$ be the time to compute $F(n)$ by dynamic programming. The recurrence is $D(n) = D(n-1) + 1$.
- The solution, by the anti-derivative method, is $D(n) = \Theta(n)$. Of course, the answer is obviously $\Theta(n)$, which you can see without using a recurrence.

(d) [10 points] If $F(n)$ is computed using memoization, how many memos are created? (Asymptotic answer, please.)

- The only values of $i$ for which $F(i)$ will be computed are $i = n/2^k$ for some integer $k$ such that $n/2^k \geq 1$. There are $\Theta(\log n)$ choices of $k$; thus $\Theta(\log n)$ memos will be created.
7. [20 points] Walk through Dijkstra’s Algorithm for the following weighted directed graph, where the start vertex is $s$.

![Graph Image]

8. [20 points] Consider the weighted directed graph shown below in figure (a). If you use Johnson’s algorithm, you will find adjusted weights for each edge. Label the edges in figure (b) with those adjusted weights. Do not finish executing the algorithm.

![Graph Images]

9. [20 points] What is the Levenshtein distance between $abcdef$ and $ackdeg$? Show the matrix.

The Levenshtein distance is 3.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>g</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
10. [20 points] Walk through the A* Algorithm for the weighted graph shown below to find the shortest path from S to T. The heuristic is indicated by a red numeral at each vertex.