1. True or False. Write "O" if the answer is not known to science at this time. [5 points each]

(a) ______ All sparse graphs are planar.
(b) ______ The time complexity of quicksort is $O(n^2)$.
(c) ______ The time complexity of quicksort is $\Omega(n \log n)$.
(d) ______ If a problem is $NP$-complete, there is no polynomial time algorithm which solves it.

2. Fill in the blanks. [5 points each blank.]

(a) __________________________ algorithm for solving the single source shortest path problem only works correctly if there are no negative weights.

(b) In dynamic programming the subproblems must be worked in __________________________ order.

(c) Vertices $u$ and $v$ of a graph $G$ belong to different __________________________ if there is no path in $G$ from $u$ to $v$.

(d) If there is no directed path in a directed graph from vertex $u$ to vertex $v$, then $u$ and $v$ belong to different __________________________

(e) An $n \times n$ matrix with $n \log n$ non-zero entries would probably be considered __________________________
3. Solve the recurrences. Give asymptotic answers in terms of \( n \), using either \( O \), \( \Omega \), or \( \Theta \), whichever is most appropriate. (5 points each)

(a) \( F(n) = 2F\left(\frac{n}{2}\right) + n \)

(b) \( F(n) \geq 4F\left(\frac{n}{2}\right) + n^2 \)

(c) \( F(n) = F(n - 1) + \frac{n}{4} \)

(d) \( F(n) = F(n - \sqrt{n}) + \sqrt{n} \)

(e) \( F(n) = F(\log n) + 1 \)

(f) \( T(n) < T(n - 2) + n^2 \)

(g) \( G(n) \geq G(n - 1) + n \)

(h) \( H(n) \leq 2H(\sqrt{n}) + O(\log n) \).

(i) \( F(n) = F(n/2) + 1 \)

(j) \( F(n) = F(n - 1) + O(\log n) \)

(k) \( F(n) = F\left(\frac{n}{2}\right) + 2F\left(\frac{n}{4}\right) + n \)

(l) \( F(n) = F\left(\frac{3n}{5}\right) + F\left(\frac{4n}{5}\right) + n^2 \)
4. Find the asymptotic complexity, in terms of \( n \), for each of these fragments, expressing the answers using \( O \), \( \Theta \), or \( \Omega \), whichever is most appropriate. (5 points each)

(a) \( \text{for(int } i = 0; i < n; i = i+1) \)

(b) \( \text{for(int } i = 1; i < n; i = 2*i) \)

(c) \( \text{for(int } i = 2; i < n; i = i*i) \)

(d) \( \text{for(int } i = n; i > 0; i--) \)
\( \quad \text{for(int } j = i; 2*j <= n; j = 2*j) \)

(e) \( \text{for(int } i = n; i > 0; i--) \)
\( \quad \text{for(int } j = 1; 2*j <= i; j = 2*j) \)

(f) \( \text{for(int } i = 1; i < n; i++) \)
\( \quad \text{for(int } j = n; j > 0; j = j/2) \)

(g) \( \text{for(int } i = n; i > 1; i = \sqrt{i}) \)

(h) \( \text{for(int } i = 1; i*i < n; i++) \)

5. [10 points each] Give the asymptotic time complexity, in terms of \( n \), for each of these recursive subpro-grams.

(a) \( \text{int } f(\text{int } n) \)
\( \quad \{ \)
\( \quad \quad \text{if (} n < 2 \text{) return 1; } \)
\( \quad \quad \text{else return } f(n-1)+f(n-1); \)
\( \quad \} \)

(b) \( \text{void hello(\text{int } n)} \)
\( \quad \{ \)
\( \quad \quad \text{if(} n >= 1 \text{)} \)
\( \quad \quad \quad \{ \)
\( \quad \quad \quad \quad \text{for(int } i = 1; i < n; i++) \)
\( \quad \quad \quad \quad \quad \text{cout } << \text{"Hello!" } << \text{endl; } \)
\( \quad \quad \quad \quad \text{hello(n/2); } \)
\( \quad \quad \quad \text{hello(n/2); } \)
\( \quad \quad \} \)
\( \} \)
6. [30 points] Which of the following problems are known to be NP-complete? Mark T or F.

(a) ______ Given a weighted graph and a number $B$, does the graph have a Hamiltonian cycle of weight at most $B$?

(b) ______ Given a table and a set of tiles of various shapes, can the tiles all be placed on the table so that none overlap and none overhang the edge?

(c) ______ Given a weighted graph, a number $D$, and two vertices $u$ and $v$, does there exist a path between $u$ and $v$ of weight at most $D$?

7. [20 points] What is the name of the algorithm implemented by the following code?

```c
int x[n];
read in values of x from some external source;
for(int i = 0; i < n; i++)
  for(j = i+1, j < n; j++)
    if(x[j] < x[i]) swap(x[i],x[j]);
```

8. [20 points] A directed graph can be represented in the computer in several ways. Two of them are an array of out-neighbor lists, and an array of in-neighbor lists. Let $G$ be a directed graph whose vertices are the integers $0 \ldots 19$, and whose array of out-neighbor lists is as written below. Construct the array of in-neighbor lists of $G$.

<table>
<thead>
<tr>
<th>Out-neighbor lists:</th>
<th>In-neighbor lists:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 1,13</td>
<td>0:</td>
</tr>
<tr>
<td>1: 5,19</td>
<td>1:</td>
</tr>
<tr>
<td>2: 5,14</td>
<td>2:</td>
</tr>
<tr>
<td>3: 9</td>
<td>3:</td>
</tr>
<tr>
<td>4: 7,15</td>
<td>4:</td>
</tr>
<tr>
<td>5: 4,10</td>
<td>5:</td>
</tr>
<tr>
<td>6: 11</td>
<td>6:</td>
</tr>
<tr>
<td>7: 16,18</td>
<td>7:</td>
</tr>
<tr>
<td>8: 17</td>
<td>8:</td>
</tr>
<tr>
<td>9: 8,15</td>
<td>9:</td>
</tr>
<tr>
<td>10: 4</td>
<td>10:</td>
</tr>
<tr>
<td>11: 7</td>
<td>11:</td>
</tr>
<tr>
<td>12: 2, 6</td>
<td>12:</td>
</tr>
<tr>
<td>13: 1, 2</td>
<td>13:</td>
</tr>
<tr>
<td>14: 11</td>
<td>14:</td>
</tr>
<tr>
<td>15: 17,18</td>
<td>15:</td>
</tr>
<tr>
<td>16:</td>
<td>16:</td>
</tr>
<tr>
<td>17:</td>
<td>17:</td>
</tr>
<tr>
<td>18:</td>
<td>18:</td>
</tr>
<tr>
<td>19: 3, 5</td>
<td>19:</td>
</tr>
</tbody>
</table>
9. [20 points] Give pseudocode for the Floyd-Warshall algorithm. The vertices are the integers 1, 2, \ldots, n. \( W[i, j] \) is the given weight of the directed edge from \( i \) to \( j \), which is \( \infty \) if there is no edge from \( i \) to \( j \). The output of the algorithm is \( F[i, j] \) for all \( i \) and \( j \), the minimum length of any directed path from \( i \) to \( j \), as well as the backpointer \( B[i, j] \) for all \( i \neq j \). Assume the graph is strongly connected and that there are no negative cycles.
10. [20 points] Let the function $F$ on positive integers be defined as follows:

$$F(n) = \begin{cases} 
1 & \text{if } n \leq 2 \\
(F(\frac{n}{2}) \ast (F(\frac{n-2}{2}) + F(\frac{n+2}{2})) \mod 23 & \text{if } n > 2 \text{ and } n \text{ is even} \\
(F(\frac{n-1}{2})^2 + (F(\frac{n+1}{2})^2) \mod 23 & \text{if } n > 2 \text{ and } n \text{ is odd}
\end{cases}$$

You can compute $F(n)$ for any $n$ in $O(n)$ time using dynamic programming. However, the computation can be done much faster by using memoization.

Write pseudo-code for a memoization algorithm which prints $F(n)$ for some given $n$ in less than linear time.
11.  [20 points] Use the DFS-based algorithm in our textbook to find the strong components of the directed graph $G$ shown below. Of course, you can easily “eyeball” the answer, but I want to see the steps of the algorithm.
12. [20 points] Find the longest increasing subsequence of the sequence 5, 1, 2, 7, 0, 9, 3, 5, 8, using the dynamic programming algorithm I showed you in the video. Show all work.

13. [20 points] This problem requires serious thought. Consider the following code:

```java
for(int i = 2; i < n*n; i = i*i)
    for(int j = 1; j < i; j++)
```

The asymptotic time complexity of this code is not \( \Theta \) of any of the functions of \( n \) listed at the beginning of the test, but it is \( O \) of one of those functions, and is also \( \Omega \) of another one of those functions. Find the two functions. (Hint: When you are faced with the problem of finding an unknown formula, it frequently helps to experiment by choosing some test numbers. Even bigger hint: try \( n = 16, 17, 256, \) and 257.)