1. True or False. Write “O” if the answer is not known to science at this time. [5 points each]
   
   (a) F Computers are so fast today that complexity theory is only of theoretical, but not practical, interest.
   
   (b) T The inverse Ackermann function, \( \alpha(n) \), grows so slowly that, from a practical (as opposed to theoretical) point of view, it might as well be constant.
   
   (c) O If a problem is \( \mathcal{NP} \)-complete, there is no polynomial time algorithm which solves it.
   
   (d) F Quicksort takes \( \Theta(n \log n) \) time on an array of size \( n \).
   
   (e) T Planar graphs are sparse.
   
   (f) T Acyclic graphs are sparse.
   
   (g) F Acyclic directed graphs are sparse.

2. Fill in the blanks. [5 points each blank.]
   
   (a) If a planar graph \( G \) has 20 edges, then the number of vertices of \( G \) cannot be less than 9.
   
   (b) A directed acyclic graph with 5 vertices cannot have more than 10 arcs, and a directed acyclic graph with 6 vertices cannot have more than 15 arcs. A directed acyclic graph with 10 vertices cannot have more than 45 arcs.
   
   (c) A directed acyclic graph with 20 arcs cannot have fewer than 7 vertices. (You must give the best possible answer, exactly. No partial credit.)
   
   (d) The height of a binary tree with 50 nodes is at least 5.
   
   (e) In perfect hashing, there are no collisions.
   
   (f) If separate chaining is used to resolve collisions in a hash table with \( n \) items and \( n \) places in the array and if the hash function is pseudo-random, then approximately 8% of the places will have more than two items. Pick the best answer from among these choices: (0%, 1%, 2%, 4%, 8%, 16%, 32%)
   
   Hint: approximately 36.8% of the places will have no items.
   
   (g) \( \Omega(n \log n) \).
   
   (h) Radix, or bucket sorting is not comparison-based.
   
   (i) The infix expression \((x + y) \ast z\) is equivalent to the prefix expression \( \ast + xyz \) and the postfix expression \( xy + z\ast \).
(j) What is the **only** difference between the abstract data types `queue` and `stack`?

In a stack, only the most recently inserted item may be deleted (LIFO), while in a queue, only the least recently inserted item may be deleted (LIFO).

(k) The items stored in a priority queue (that includes stacks, queues, and heaps) represent **unfulfilled obligations**.

(l) Name a divide-and-conquer searching algorithm.

**Binary search.**

(m) Name two divide-and-conquer sorting algorithms.

**Mergesort** and **quicksort**.

(n) The following is pseudo-code for which sorting algorithm we’ve discussed?

**Selection sort.**

```c
int x[n];
obtain values of x;
for(int i = n-1; i > 0; i--)
    Find the largest element of x[0], ... x[i] and swap it with x[i]
```

(o) The following is pseudo-code for which sorting algorithm we’ve discussed?

**Bubblesort.**

```c
int x[n];
obtain values of x;
bool finished = false;
for(int i = n-1; i > 0 and not finished; i--)
{
    finished = true;
    for(int j = 0; j < i; j++)
        if(x[j] > x[j+1])
        {
            swap(x[j],x[j+1]);
            finished = false;
        }
}
```

3. Give the asymptotic complexity, in terms of $n$, of each of the following code fragments. [10 points each]

(a) `for(int i = n; i > 1; i = i/2)`

`cout << "hello world" << endl;`

$\Theta(\log n)$

(b) `for(int i = 1; i < n; i++)`

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

`cout << "hello world" << endl;`

$\Theta(n \log n)$
(c) for(int i = 1; i < n; i++)
    for(int j = i; j < n; j = 2*j)
        cout << "hello world" << endl;

Θ(n)

(d) for(int i = 2; i < n; i = i*i)
    cout << "hello world" << endl;

Θ(log log n)

4. [10 points] Name one problem which is known to be \( \mathcal{NP} \)-complete. ________________________________

5. Solve the recurrences. Give asymptotic answers in terms of \( n \), using either \( O \), \( \Omega \), or \( \Theta \), whichever is most appropriate. (10 points each)

| (a) \( F(n) = 2F\left(\frac{n}{2}\right) + n \) | \( \Theta(n \log n) \) |
| (b) \( F(n) \geq 4F\left(\frac{n}{2}\right) + n^2 \) | \( \Theta(n^2 \log n) \) |
| (c) \( F(n) = F(n-1) + \frac{n}{4} \) | \( \Theta(n^2) \) |
| (d) \( F(n) \leq F\left(\frac{n}{2}\right) + F\left(\frac{n}{4}\right) + F\left(\frac{n}{8}\right) + n \) | \( \Theta(n) \) |
| (e) \( F(n) = F(n - \sqrt{n}) + n \) | \( \Theta(n^{3/2}) \) |
| (f) \( F(n) = F(\log n) + 1 \) | \( \Theta(\log^* n) \) |

6. [20 points] Use dynamic programming to compute the length of the longest common subsequence of the strings “011011001” and “1010011001.”

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

The length of the longest common subsequence is 8.
7. [20 points] Use dynamic programming to compute the Levenshtein distance between the strings “abcd-abc” and “bdabcd.”

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>a</th>
<th>b</th>
<th>c</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>
<td>7</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

The Levenshtein distance between those two strings is 4.

8. [20 points] Design a dynamic programming to compute the maximum sum of any contiguous subsequence of a given sequence of numbers. For example, if the given sequence is 2, 1, −4, 6, = 3, 7, −1.2, −2, 1 that sum is 6 + (−3) + 7 + (−1) + 2 = 11. (There is an $O(n)$-time algorithm.)

Let $x_1, x_2, \ldots, x_n$ be the sequence. We construct two arrays. Let $A[i]$ be the largest sum of any contiguous subsequence of $x_1, x_2, \ldots, x_i$, and let $B[i]$ be the largest sum of any contiguous subsequence of $x_1, x_2, \ldots, x_i$ which includes $x_i$. Write $X[i]$ for $x_i$.

$$A[0] = 0;$$
$$A[1] = \max\{0, B[1]\}$$

for (i = 2 to n)
{
    $B[i] = X[i] + \max\{0, B[i-1]\}$
    $A[i] = \max\{A[i-1], B[i]\}$
}

write $A[n]$.

9. Solve each of the following recurrences, giving the answer in terms of $O$, $\Theta$, or $\Omega$, whichever is most appropriate [10 points each].

(a) $T(n) < T(n-2) + n^2$  \hspace{1cm} $T(n) = O(n^3)$
(b) $F(n) \geq F(\sqrt{n}) + \lg n$  \hspace{1cm} $F(n) = \Omega(\log n)$
(c) $G(n) \geq G(n-1) + n$  \hspace{1cm} $G(n) = \Omega(n^2)$
(d) $F(n) = 4F(n/2) + n^2$  \hspace{1cm} $F(n) = \Theta(n^2)$
(e) $H(n) \leq 2H(\sqrt{n}) + O(\log n)$  \hspace{1cm} $H(n) = O(\log n \log \log n)$
(f) $K(n) = K(n - \sqrt{n}) + 1$  \hspace{1cm} $K(n) = \Theta(\sqrt{n})$
(g) $F(n) = 4F\left(\frac{3n}{4}\right) + n^5$  \hspace{1cm} $F(n) = \Theta(n^5)$
10. 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.

(a) for(i = 0; i < n; i = i+1);
cout << "Hi!" << endl;
$\Theta(n)$

(b) for(i = 1; i < n; i = 2*i);
cout << "Hi!" << endl;
$\Theta(\log n)$

(c) for(i = 2; i < n; i = i*i);
cout << "Hi!" << endl;
$\Theta(\log \log n)$

(d) The following code models the first phase of heapsort.
for(int i = n; i > 0; i--)
for(int j = i; 2*j <= n; j = 2*j)
cout << "swap" << endl;
$\Theta(n)$

(e) The following code models the second phase of heapsort.
for(int i = n; i > 0; i--)
{
    cout << "swap" << endl;
    for(int j = 1; 2*j <= i; j = 2*j)
        cout << "swap" << endl;
}
$\Theta(n \log n)$

(f) The following code models insertion of $n$ items into an AVL tree.
for(int i = 1; i < n; i++)
for(int j = n; j > 0; j = j/2)
    cout << "check AVL property and possibly rotate" << endl;
$\Theta(n \log n)$

11. Solve each of the following recurrences, expressing the answers using $O$, $\Theta$, or $\Omega$, whichever is most appropriate. [10 points each]

(a) $F(n) = F(n/2) + 1$  \quad $F(n) = \Theta(\log n)$

(b) $F(n) = F(n - 1) + O(\log n)$

(c) $F(n) = F\left(\frac{n}{2}\right) + 2F\left(\frac{n}{4}\right) + n$  \quad $F(n) = \Theta(n \log n)$

(d) $F(n) = F\left(\frac{3n}{5}\right) + F\left(\frac{4n}{5}\right) + n^2$  \quad $F(n) = \Theta(n^2 \log n)$

(e) $F(n) = F(n - 2) + n$  \quad $F(n) = \Theta(n^2)$
12. Use Huffman's algorithm to construct an optimal prefix code for the alphabet \{A, B, C, D, E, F\} where the frequencies of the symbols are given by the following table.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>7</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
</tr>
</tbody>
</table>


Assume that \{X[i]\} for 0 \leq i < n is an ordered array. Let x be the sought value.

```cpp
lo = 0
hi = n
while(lo+1 < hi)
{
    mid = (lo+hi)/2
    if(x < X[mid]) hi = mid
    else lo = mid
}
if(x = X[hi]) write "x is found in position hi"
else write "x is not in the array"
```

14. 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. [10 points each]

(a) for(int i = 1; i*i < n; i++)
    cout << "Hi!" << endl;
\(\Theta(\sqrt{n})\)

(b) for(int i = n; i > 1; i = sqrt(i));
    cout << "Hi!" << endl;
\(\Theta(\log \log n)\)

Find the asymptotic time complexity, in terms of \(n\), for each of these functions, expressing the answers using \(O\), \(\Theta\), or \(\Omega\), whichever is most appropriate. [10 points each]

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

(b) void hello(int n)
    {

if (n >= 1)
{
    for (int i = 1; i < n; i++)
        cout << "Hello!" << endl;
    hello(n/2);
    hello(n/2);
}

Θ(n log n)

15. [20 points] Define the Collatz function as follows:

```cpp
int collatz(int n)
{
    assert(n > 0);
    if (n == 1) return 0;
    else if (n%2) return collatz(3*n+1); // n is odd, greater than 1
    else return collatz(n/2); // n is even
}
```

Write pseudo-code for a memoization algorithm which prints collatz(n) for all n from 1 to 1000.

16. [20 points] Give pseudocode for a recursive algorithm which computes the median of the union of two sorted lists in logarithmic time.

17. [20 points] Describe a randomized algorithm which finds the kth smallest element of an unsorted list of n distinct numbers, for a given k ≤ n, in O(n) expected time. (By “distinct,” I mean that no two numbers in the list are equal.)

18. [20 points] Walk through the A* algorithm for the following weighted graph to find the shortest path from S to T. Edge weights are shown in black, and the values of the heuristic are shown in red.

19. [20 points] Circle the strong components of the directed graph.


```java
for(int i = 0; i < n; i++)
    for(int j = 0; j < n; j++)
        dist[i,j] = infinity;

for(int i = 0; i < n; i++)
    dist[i,i] = 0;

for(int j = 0; j < n; j++)
    for(int i = 0; i < n; i++)
        for(int k = 0; k < n; k++)
            
            temp = dist[i,j] + dist[j,k];
            if(temp < dist[i,k])
                
                dist[i,k] = temp;
                back[i,k] = i;
          }
```

22. [20 points] Show the minimum spanning tree of the following weighted graph.