1. Solve the recurrences. Give asymptotic answers in terms of $n$, using either $O$, $\Omega$, or $\Theta$, whichever is most appropriate. Use whichever technique is appropriate for each problem. [10 points each]

(a) $F(n) = 2F(n/2) + n$ 
$F(n) = \Theta(n \log n)$ by the master theorem.

(b) $F(n) \leq F(n-3) + 3 \log n$ 
$F(n) - F(n-3) = 3 \log n$ 
$F(n) = \Theta(\log n)$

(c) $F(n) = F(\sqrt{n}) + 1$; (Hint: use a substitution. Introduce the variable $m$ and let $m = \log_2 n$, and introduce the function $G$ such that $G(m) = F(2^m) = F(n)$. Then find a new recurrence using the function $G$. Solve that recurrence, and then substitute back to solve the original recurrence.)

$G(m) = G(m/2) + 1$ 
$F(n) = G(m) = \Theta(\log m)$ by the master theorem

(d) $F(n) \leq F(n/2) + F(n/3) + n$ 
$F(n) = O(n)$ by the generalized master theorem since $\frac{1}{2} + \frac{1}{3} < 1$

(e) $F(n) \leq 2F(n-1) + 1$ 
Let $m = 2^n$ and $G(m) = F(n)$. Then $m/2 = 2^{n-1}$ hence $F(n-1) = G(m/2)$

$G(m) = 2G(m/2) + 1$ 
$G(m) = \Theta(m)$ by the master theorem

$F(n) = G(m) = \Theta(2^n)$

(f) $F(n) \geq 4F(n/2) + n$ 
$F(n) = \Omega(n^2)$ by the master theorem since $\log_2 4 = 2$

(g) $F(n) = F(n - \sqrt{n}) + \sqrt{n}$ 
$F(n) - F(n - \sqrt{n}) = \sqrt{n}$ 
$F(n) = \Theta(1)$

2. [10 points] Find an integer $K$ such that the solution to the recurrence below is $F(n) = \Theta(n^2 \log n)$.

$F(n) = F(3n/5) + K F(n/5) + n^2$

By the master theorem, 
$\left(\frac{3}{5}\right)^2 + K \left(\frac{1}{5}\right)^2 = 1$ hence $K = 16$. 


3. Give the asymptotic time complexity of each of these code fragments, in terms of \( n \). [10 points each.]

(a) `for(int i = n; i > 1; i = log2(i))`

Recall the recursive definition: \( \log^* x = \begin{cases} 0 & \text{if } x \leq 1 \\ 1 + \log^* \log_2 x & \text{otherwise} \end{cases} \)

Thus \( \log^* \log_2 x = \log^* x - 1 \) if \( x > 1 \).

Let \( j = \log^* i \). We obtain

`for(int j = log^*n; j > 1; j = j-1)`

The answer is then \( \Theta(\log^* n) \).

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

`for(int j = 1; j < i; j++)`

Let \( k = \log_2 i \): then \( i = 2^k \). We obtain

`for(int k = log2(n); k > 0; k = k-1)`

`for(int j = 1; j < 2^k; j++)`

We approximating by integrals, letting \( y, z \) be the continuous analogs of \( j, k \), respectively. Conventional notation of integrals requires that the values of \( z \) must be increasing despite the fact that the values of \( k \) are decreasing in the code.

\[
\int_{z=1}^{\log_2 n} \int_{y=1}^{2^z} dydz = \int_{z=1}^{\log_2 n} (2^z - 1)dz = \left( \frac{2^z}{\ln 2} - z \right)_{z=1}^{\log_2 n} = \Theta(n)
\]

Alternative computation using summations. Conventional notation requires the value of the index \( k \) to increase, despite the fact that it decreases in the code. Recall that \( 1 + 2 + 4 + \cdots + 2^m = 2^{m+1} - 1 \).

\[
\sum_{k=0}^{\log_2 n - 1} 2^k = 2^{\log_2 n} - 1 = n - 1 = \Theta(n)
\]

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

`for(int j = i; j < n; j++)`

Using the same substitution \( k = \log_2 i \) and using the summation method:

\[
\sum_{k=0}^{\log_2 n - 1} (n - 2^k) = n(\log_2 n) - 2^{\log_2 n} = kn - (n - 1) = \Theta(n \log n)
\]

(d) `for(int i = 0; i < n; i++)`

`for(int j = 0; j < i*i; j++)`

Recall that \( 1 + 4 + 9 + \cdots m^2 = \frac{m(m + 1)(2m + 1)}{6} \)

\[
\sum_{i=0}^{n-1} i^2 = \Theta(n^3)
\]
We can use integration instead:

\[
\sum_{i=0}^{n-1} i^2 = \Theta \left( \int_{x=0}^{n-1} x^2 \, dx \right) = \Theta \left( \frac{(n-1)^3}{3} \right) = \Theta(n^3)
\]

(e) for(int i = 2; i < n; i = i*i)

Let \( j = \log_2 i \)

for(int j = 1; j < \log2(n); j = 2*j)

Let \( k = \log_2 j \)

for(int k = 0; k < \log2(\log2(n)); k = k+1)

\( \Theta(\log \log n) \)

(f) for(int i = n; i > 1; i = sqrt(i))

Let \( j = \log_2 i \)

for(int j = \log2(n); j > 0; j = j/2)

Let \( k = \log_2 j \)

for(int k = \log2(\log2(n)); k >= 0; k = k-1)

\( \Theta(\log \log n) \)

(g) for(int i = n; i > 1; i = sqrt(i))

for(int ell = 0; ell < i; ell++)

Since I want to use \( j \) for \( \log_2 i \), I will change the second variable to \( \ell \). Let \( j = \log_2 i \)

for(int j = \log2(n); j > 0; j = j/2)

for(int ell = 0; ell < 2^j; ell++)

Let \( k = \log_2 j \)

for(int k = \log2(\log2(n)); k >= 0; k = k-1)

for(int ell = 0; ell < 2^(2^k); ell++)

Use the summation method. We have \( \sum_{k=0}^{\log_2 \log_2 n} 2^{2^k} \)

The last term of the summation is greater than the sum of all the previous terms, thus the summation is \( \Theta \left( 2^{2^{\log_2 \log_2 n}} \right) = \Theta(n) \).
4. Arrays. There is a built-in implementation of the abstract data type Array in C++. However, sometimes it is best to use a different implementation. In Problems 4 and 5 I was very generous in grading, since I realized the answers were not clear-cut. I won’t make that mistake on the final.

Suppose that an \( n \times m \) array \( A \) has \( k \) non-zero entries. I wanted you to choose one of the following implementations:

- Standard implementations: one memory location for each entry.
- Array of search structures, one for each row, an array \( R \) of length \( n \), where \( R[i] \) is a search structure which holds all ordered pairs of the form \((j, x)\) where \( A[i][j] = x \) and \( x \neq 0 \).
- Search structure of non-zero entries: A search structure which holds all ordered triples \((i, j, x)\) where \( A[i][j] = x \) and \( x \neq 0 \).

(a) [10 points] What space saving data structure would you use to implement a \( 1000 \times 1000 \) 2-dimensional array where there are 2000 non-zero entries, the rest zero?

Use an array of search structures, one for the non-zero entries of each row.

(b) [10 points] What space saving data structure would you use to implement a \( 10000 \times 10000 \) 2-dimensional array which has 100 non-zero entries, the rest zero?

Search structure of non-zero entries.

5. Graphs.

(a) [10 points] What does it mean to say that a graph is “sparse”?

The term is not well-defined, but generally it means that the number of edges in the graph is much lower than the maximum possible. If \( n, m \) are the numbers of vertices and edges, we can usually say the graph is sparse if \( m = o(n^2) \).

(b) [10 points] What data structure would you use to implement a graph with 1000 vertices and 10000 edges?

The answer I want is, an array of search structures, one for each vertex. The entry for a vertex \( v \) is a search structure containing all neighbors of \( v \).

(c) [5 points] What is the maximum number of edges a planar graph with 5 vertices can have?

The formula for a planar graph is \( m \leq 3n - 6 \) if \( n > 2 \). Thus \( m \leq 9 \).
6. [20 points] Construct an optimal prefix-free code for the alphabet a,b,c,d,e,f with the frequencies given by the table below.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>12</td>
<td>101</td>
</tr>
<tr>
<td>b</td>
<td>6</td>
<td>000</td>
</tr>
<tr>
<td>c</td>
<td>7</td>
<td>001</td>
</tr>
<tr>
<td>d</td>
<td>15</td>
<td>01</td>
</tr>
<tr>
<td>e</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>f</td>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>g</td>
<td>5</td>
<td>1001</td>
</tr>
</tbody>
</table>

This answer is not unique. Any prefix-free code where the code strings for d and e have length 2, for a, b, and c have length 3, and for f and g have length 4, is optimal.

7. [20 points] In my video at https://www.youtube.com/watch?v=iKA4URlAtKo I show how to implement a min-heap as an array. For example, a min-heap of size 6 could be implemented as the following array:

```
D F H M L R
```

If `deletemin` is executed, the array changes in a sequence of steps. Show those steps.

```
R F H M L R
F R H M L R
F L H M R R
```

Subsequently, E is inserted. Show the array after each step.

```
F L H M R E
F L E M R H
E L F M R H
```

8. [20 points] Explain, using diagrams and text, but not pseudo-code, the linked list implementation of a stack. Represent the items on the stack by capital letters.

(a) Illustrate the stack with item K, F, L, A, in that order, where K is the top item.

(b) Starting with the previous stack, illustrate `pop`.

(c) Starting with the previous stack, illustrate `push`, where the item D is pushed onto the stack.
9. Let $G = (V, E)$ be a weighted directed graph with $n$ vertices and $m$ edges. The vertices are $v_1, v_2, \ldots, v_n$ and the edges are $e_1, e_2, \ldots, e_m$.

Each $e_j$ is a directed edge $(x_j, y_j)$, where $x_j, y_j \in V$. $W[e_j] = W[x_j, y_j]$ is the given weight of the edge $e_j$. Note that a vertex could have several names. For example, If $e_3$ is a directed edge from $v_2$ to $v_5$, $v_2$ has the alternative name $x_3$ and $v_5$ has the alternative name $y_3$, and $W[e_3] = W[x_3, y_3] = W[v_2, v_5]$.

Here is pseudo-code for the Bellman-Ford algorithm for the single source shortest path problem for $G$, where $v_1$ is the source. We will write $F[v_i]$ for the smallest weight of any path from $v_1$ to $v_i$ that we have found so far, and we write $B[v_i]$ for the backpointer of that path. The arrays $F$ and $B$ are the outputs of the program.

(a) [10 points] Fill in the missing line which assigns a backpointer.

(b) [20 points] The main outer loop iterates $n - 1$ times. However, in most practical situations, The values of $F$ are updated only during the first few iterations. Insert code (4 lines) to end the outer loop if there are no further changes to $F$. Assume that $G$ has no negative cycle.

\[
F[v_1] = 0;  \\
\text{for all } i \text{ from } 2 \text{ to } n  \\
\quad F[v_i] = \infty;  \\
\text{bool } \text{changed} = \text{true};  \\
\text{for all } t \text{ from } 1 \text{ to } n - 1  \\
\quad \text{if(}\text{changed}\text{)}  \\
\quad \quad \text{changed} = \text{false};  \\
\quad \text{for all } j \text{ from } 1 \text{ to } m  \\
\quad \quad \text{if } (F[x_j] + W[e_j] < F[y_j])  \\
\quad \quad \quad \{  \\
\quad \quad \quad \quad F[y_j] = F[x_j] + W[e_j];  \\
\quad \quad \quad \quad B[y_j] = x_j \text{ //assign a backpointer}  \\
\quad \quad \quad \quad \text{changed} = \text{true};  \\
\quad \quad \}  \\
\}  \\
\]