2. Binary Search is $\Theta(\log_2 n)$, so $\lceil \log_2 700,000,000 \rceil = 30$ recursive calls, but 2 compares per call so 60 compares in worst case.

7. int findLargest (int list[], int low, int high) 
   
   int mid;
   
   if (low == high) {
     return list[low];
   }
   
   else 
   
   mid = (low + high)/2;
   
   leftLargest = findLargest (list, low, mid);
   
   rightLargest = findLargest (list, mid +1, high);
   
   if (leftLargest > rightLargest)
     return leftLargest;
   else
     return rightLargest;
Analysis of findLargest

Using recurrence and Master Thm.

\[ W(n) = 2W\left(\frac{n}{2}\right) + 1 \quad n^0 \quad a=2, b=2, k=0 \]

\[ W(1) = 0 \]

\[ a \geq b^k \]

\[ 2 > 2^0 = 1 \quad \text{use } \Theta(n \log_b a) \]

\[ \Theta(n \log_2 2) = \Theta(n) \]

---

Using a "tracing"

<table>
<thead>
<tr>
<th># compares</th>
<th># items</th>
<th>divide steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>n</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>n/2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>n/4</td>
<td>4</td>
</tr>
<tr>
<td>n/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n/4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \frac{t}{2n-1} \in \Theta(n) \]

HW2-2
8. Trace merge sort

recursion
divide steps

combine by merging

16. \( T(n) = 10T\left(\frac{n}{3}\right) + cn^2 \)
   
   \( a = 10 \)  \( b = 3 \)  \( k = 2 \)

\( q \?
\)

\( 10 > 3^2 = 9 \) use \( \Theta(n^{\log_b q}) \)

\( \leq \Theta(n^{\log_3 10}) = \Theta(n^{2.095}) \)
19. Trace Quicksort

\[ 123 \quad 34 \quad 189 \quad 56 \quad 150 \quad 12 \quad 9 \quad 240 \]

pivot item

\[ 9 \quad 34 \quad 56 \quad 12 \]

pivot item

\[ 9 \quad 12 \quad 34 \quad 56 \quad 123 \quad 150 \quad 189 \quad 240 \]
24. Quicksort

(a) worst-case is descending order which causes the most compares and moves
   e.g. 10 9 8 7 6 5 4 3 2 1

(b) best-case pivot always falls in the middle
   e.g. 6 4 5 2 1 3 8 7 10 9 11

40. Some people misunderstood the ordering "\( T[i][j] \leq T[i][j+1] \)"
   implies each row is in ascending (non-decreasing actually) order.
   "\( T[i][j] \leq T[i+1][j] \)" implies each column is in non-decreasing order as
   you scan it from top-to-bottom.
For example the following Table satisfies this ordering:

Table:

\[
\begin{bmatrix}
5 & 10 & 20 & 70 & 100 \\
10 & 11 & 25 & 80 & 110 \\
20 & 27 & 90 & 100 & 115 \\
200 & 210 & 220 & 240 & 300
\end{bmatrix}
\]

Since the chapter deals with divide-and-conquer, you need a divide-and-conquer solution. I will model my algorithm after binary search by comparing the Target item to the middle item of the 2D Table.

If Target equals the middle item, return its row and column.
If Target > middle item, we can eliminate the "upper left" quarter of Table

![Diagram of a table with a target of 200 and a range of values]

We can then search the remaining Table by splitting it into three smaller searches. If we find the Target in any of these, we return its location; otherwise, return [-1, -1] to indicate an unsuccessful search.

Analysis: \( W(n \times m) = 3 W \left( \frac{n \times m}{4} \right) + c (n \times m)^{\frac{9}{10}} \)

\( q \leq k \)

\( 3 > 4^0 \) so use \( \Theta(n^{\log_69}) = \Theta(n^{\log_43}) = \Theta(n^{0.79}) \)