1. One day, one of your friends challenges you to play a game which requires you to guess a number, within the rage of 1 to 100, that he has set in mind. What is the minimum number of guesses that you can guarantee to get the right answer? Please prove your approach. [hint: Binary Search]
2. Please define what is a stable sorting algorithm and an unstable sorting algorithm, and then give an example to each type of sorting algorithm.
3. (a) Show the step-by-step insertion sort result of the following list, [5, 6, 3, 8, 2].

(b) Show the step-by-step merge sort result of the following list, [50, 10, 90, 30, 70, 40, 80, 60, 20]

1. Suppose that we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the following resulted array:  
   2 5 1 7 9 12 11 10

Among the following statements which one is correct? Explain why.

1. The pivot could be either the array element containing 7 or 9.
2. The pivot could be the array element containing 7, but not 9.
3. The pivot is not the array element containing 7, but the one of 9.
4. Neither the array element containing 7 nor 9 is the pivot.
5. Let P be a Quick Sort Program to sort numbers into ascending order using the first element as pivot. Suppose that the total number of comparisons is t1 if apply P to the inputs {1, 2, 3, 4, 5}, and is t2 if apply P to {4, 1, 5, 3, 2}. Then which of the following shall hold?
6. t1=5 (b) t1<t2 (c) t1>t2 (d) t1=t2
7. Please prove that under the limitation of comparison and interchange, the best time complexity of sorting algorithm is .
8. Quicksort is a fast and commonly used comparison-based sorting algorithm. Please refer to the following quicksort pseudocode and answer questions listed after.

QUICKSORT(A,p,r)

1 if p<r

2 q=PARTITION(A,p,r)

3 QUICKSORT(A,p,q-1)

4 QUICKSORT(A,q+1,r)

PARTITION(A,p,r)

1 x=A[r]

2 i=p-1

3 for j=p to r-1

4 if A[j]<=x

5 i=i+1

6 exchange A[i] with A[j]

7 exchange A[i+1] with A[r]

8 return i+1

(a) Answer true or false :

(1) The output of QUICKSORT() above is a **non-increasing** sorted sequence.

(2) The QUICKSORT() above is a **stable** sorting algorithm.

(3) The QUICKSORT() above is an **in-place** sorting algorithm.

(b) The given version of QUICKSORT above cannot efficiently process the cases in which the input array has a large number of repeated elements. Analyze the worst-case running time which occurs when all *n* keys in the input array A are identical. Please show the complexity in the big-O notation in terms of n.

(c) To address the previously mentioned problematic case, we shall now develop a new partition algorithm PARTITIONTHREE() that separates the input keys into three groups, the ones that are smaller than, identical to, or larger than the pivot key x. The return values q1 and q2 represent the indices of the first and the last keys of the group that equals the pivot key. Please fill the blanks (A), (B), (C) in PARTITIONTHREE() below to complete the implementation, such that QUICKSORTTHREE() runs in O(n)-time when the input array has all n elements identical.

QUICKSORTTHREE(A, p, r)

1 if p<r

2 (q1,q2)=PARTITIONTHREE(A,p,r)

3 QUICKSORTTHREE(A,p,q1-1)

4 QUICKSORTTHREE(A,q2+1,r)

PARTITIONTHREE(A,p,r)

1 x=A[r]

2 q1=p

3 q2=r

4 j=p

5 while j<=q2

6 if A[j]<x

7 exchange A[j] with A[q1]

8 //(A)

9 j=j+1

10 elseif A[j]>x

11 exchange A[j] with A[q2]

12 //(B)

13 else

14 //(C)

15 return (q1,q2)

1. Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of each iteration of the **for** loop of *InserionSort* (Program 7.5).
2. Draw a figure similar to Figure 7.1 starting with the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18).
3. Show that if smaller sublists are sorted first, then the recursion in *QuickSort* can be simulated by a stack of depth O(log n).