Hints for Exam#2

1. **Oder of growth-rate functions in Big-Oh (i.e., O(?)) notation.**
   - $O(1), \quad O(\log_2 n), \quad O(n), \quad O(n \log_2 n), \quad O(n^2), \quad O(n^3), \quad O(2^n)$
   - Can you name each growth-rate?
   - Can you list name(s) of algorithms that have each running complexity?

   ➔ **Work with the summary table!**

2. **Compare sorting algorithms (based upon the topics/contents discussed in the class).**
   - best, average, worst case running complexity for each algorithm
   - For each sorting algorithm, can you answer the following questions:
     - What is/are the analogy or the analogies that we discussed?
     - Is swap() used?
     - Is it recursive?
     - Does it need extra memory space?
     - What characteristic(s) does each algorithm have?
     - Which algorithm(s) does/do use the followings?:
       - swap, shift, split (i.e., divide) and merge, enqueue and dequeue (i.e., concatenate), and use the decreasing gap size
     - What algorithm(s) are related with each other?
     - What algorithm(s) are affecting (or affected) to (or from) what algorithm(s)?
   - Other characteristic(s) discussed in class, but not asked above?

   ➔ **Work with the comparison table!**

3. **Trace each sort algorithm and demonstrate how the given integer numbers are sorted into ascending (or descending) order.**
   - You should be able to trace intermediate steps for a given sequence of integer array. Please use the same notations (or variable names) that we have used.
   - As I emphasized in the class, please don't skip any steps that you think them obvious since the computer is definitely not smarter than you are.
   - Given display of a couple of running steps (i.e., iterations), can you pinpoint what sorting algorithm is applied?

   ➔ **Work with the sequence of integers as you did in Assignment10!**

4. **Implement methods that swap two integer numbers in a given array.**
   - Do you remember the three swap() methods?
     - With a temporary variable and also with slash, slash, and backslash
     - Without a temporary variable, but with +, -, and –
     - Without a temporary variable, but with ^, ^, and ^
   - Don't forget to handle the condition that the two contents (i.e., a[i] == a[j]) are same.

Other than the aforementioned topics, you might be asked the questions similar to the followings:

(1) Suppose we are sorting an array of ten integers using a quadratic sorting algorithm. After four iterations of the algorithm's main loop, the array elements are ordered as shown here:

1 2 3 4 5 0 6 7 8 9

Then, the algorithm might be

_____.

(2) Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2 5 1 7 9 12 11 10

Then, the pivot could be

_____.

(3) Which of the following sorting algorithm does depend on whether the partitioning is balanced or unbalanced?

(A) Insertion sort (B) Selection sort (C) Quick sort (D) Merge sort

(4) Which sorting algorithm does use shift operation?

(5) What are characteristics of radix sort that are different from other sorting algorithms?

(6) Which sorting algorithm is best if the list is already sorted? Why?

(7) Which sorting algorithm is worst if the list is already sorted? Why?

(8) How to improve the original bubble sort to run in O(n) (i.e., best case)?

(9) What algorithm(s) does(do) need significant size/amount of extra memory?

(10) What algorithm(s) does(do) use shift operation?

(11) Summarize the sorting algorithms (we have discussed in the class) in terms of the following criteria.

Algorithm	Complexity (in Big O)			Is swap() used?	Is it stable?	Is it recursive?	Other notable/important characteristics?
	Best	Average	Worst				
Bubble sort							
Selection sort							
Insertion sort							
Radix sort							
Merge sort							
Quick sort							
Heap sort							

(12) You'll be asked to answer several short answer questions.