1. **(10 points)** Order the growth rates in the increasing order (i.e., fastest to slowest):
   $\boldsymbol{O(n)}$ $\boldsymbol{O(n^3)}$ $\boldsymbol{O(n\log_2 n)}$ $\boldsymbol{O(\log_2 n)}$ $\boldsymbol{O(2^n)}$ $\boldsymbol{O(1)}$ $\boldsymbol{O(n^2)}$

Complexity	Specific name	Algorithm(s)

2. **(5 points)** Write three swap() methods, each of which exchanges two integers at index ***i*** and at index ***j*** of a given array ***a[]***.

void swap1(**int []a, int i, int j**)
void swap2(**int []a, int i, int j**)
void swap3(**int []a, int i, int j**)

3. **(15 points)** Summarize the characteristics of "Sorting" algorithms that we had discussed in class.

	Complexity			**Characteristics**				
	best	avg.	worst	Swap used?	Shift used?	Extra space?	Recursive?	Other specific characteristics, analogy, etc.
bubble sort								
selection sort								
insertion sort								
shell sort								
radix sort								
merge sort								
quicksort								
peap sort								

4. **(5 points)** Trace **bubble sort** algorithm as it sorts the array **into an ascending order.**

Bottom **Top**

[**17** **9** **21** **6** **3** **12** **7**]

5. **(5 points)** Trace **selection sort** algorithm as it sorts the array **into a descending order.**

Please, select min!

[**4** **23** **10** **64** **55** **96** **7**]

6. **(5 points)** Trace **insertion sort** algorithm as it sorts the following sequence of integer array **into an ascending order.**

[**42** **23** **10** **64** **55** **96** **7**]

7. **(5 points)** Trace **radix sort** algorithm as it sorts the following sequence of integer array **into an ascending order.**

[**42** **23** **10** **64** **55** **96** **7**]

8. **(5 points)** Trace **quick sort** algorithm as it sorts the following sequence of integer array **into an ascending order.** (Assume the pivot is at the first position of the considered array.)

[**67** **90** **57** **25** **84** **32** **73** **54**]

9. **(10 points)** Trace **quick sort** algorithm as it sorts the following sequence of integer array **into an descending order.** (Assume the pivot is at the first position of the considered array.)

[**42** **23** **10** **64** **55** **37** **96** **7**]

10. **(5 points)** Trace two-way (top-down / divisive) **merge sort** algorithm as it sorts the following sequence of integer array **into a descending order.**

[**42** **23** **10** **64** **55** **37** **96** **7**]

11. **(5 points)** Trace **shell sort** algorithm that halves the gap size at each iteration as it sorts the following sequence of integer array **into an ascending order.**

[**42** **23** **10** **64** **55** **37** **96** **7**]

12. **(20 points)** Answer the questions.

(01)	[6, 8, 4, 2] [6, 8, 4, 2] What sorting algorithm is used? [6, 4, 8, 2] [6, 4, 2, 8]	
(02)	[3, 4, 2, 1, 5] [1, 4, 2, 3, 5] What sorting algorithm is used? [1, 2, 4, 3, 5]	
(03)	[4, 1, 3, 5, 2, 6] [1, 4, 3, 5, 2, 6] What sorting algorithm is used? [1, 3, 4, 5, 2, 6]	
(04)	*log(n)* often a term in execution time for divide-and-conquer algorithms, because such algorithms divide up the problems they are given, and it takes *log(n)* steps to divide a list of n elements into n lists of one element.	TRUE or FALSE
(05)	Merge sort, quicksort, and heap sort can be applicable only to lists whose length is even (i.e., power of 2).	TRUE or FALSE
(06)	The efficiency (i.e., running time) of selection sort is independent of the data being sorted.	TRUE or FALSE
(07)	Shell sort can be seen as a generalization of sorting by insertion (e.g., insertion sort) or exchange (e.g., bubble sort).	TRUE or FALSE
(08)	Given sorted lists, say [1, 2, 3, 4, 5] and [6, 7, 8, 9, 10], how many comparisons would it take to merge them?	5 / 6 / 9 / 10
(09)	Given [9, 5, 7, 6, 2] and with insertion sort, what is the first swap operation (if possible)?	9 and 5 / 5 and 7 / None
(10)	What algorithm(s) does(do) need significant size/amount of extra memory?	

13. **(15 points)** Answer the questions.

(01)	Given [1, 2, 3, 5, 4, 6], what sorting algorithm might you want to use to sort the given list? Why?	
(02)	In quicksort algorithm, picking the first element as a pivot, what problem do we run into if the given list is already ordered?	
(03)	Why are we so concern with the efficiencies of sorting algorithms?	
(04)	Assume a comparison function for complicated objects. Why might efficiency calculations for a sorting algorithm using this comparison function be misleading?	
(05)	How to improve the original bubble sort to run in *O(n)* (i.e., best case)?	