Exercise Series N°02 : Sorting Algorithms

Exercise II.1 (Bubble Sort)

Q1) Consider a list of 10 elements: $\boldsymbol{numList} = [7, 11, 3, 10, 17, 23, 1, 4, 21, 5]$.
Display the partially sorted list in **ascending** order after three complete passes of Bubble sort.

Q2) For the following array $\boldsymbol{x}$, show $\boldsymbol{x}$ after each of the first two passes of simple selection sort to arrange the elements in **ascending** order:

$\boldsymbol{i}$	0	1	2	3	4	5
$\boldsymbol{x[i]}$	30	50	70	10	40	60

Q3) **a)** For the following array $\boldsymbol{x}$, show $\boldsymbol{x}$ after each of the first two passes of bubble sort to arrange the elements in **descending** order:

$\boldsymbol{i}$	0	1	2	3	4	5
$\boldsymbol{x[i]}$	60	50	70	10	40	20

b) How many passes will bubble sort make altogether?

c) In what situations will bubble sort make the most interchanges?

Q4) Show how Bubble Sort sorts the following list $\boldsymbol{x} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D}, \boldsymbol{F}, \boldsymbol{E}, \boldsymbol{C}]$ of six items by showing the state of the list after each swap.

Exercise II.2 (Selection Sort)

Q1) Show the array, A, at the end of each iteration of the outer loop of selection sort.

Index:	0	1	2	3	4	5	6
Original array	2	1	3	8	4	6	0

Q2) Suppose Selection Sort takes $\boldsymbol{1\ Second}$ to sort $\boldsymbol{1.000\ items}$ on your new laptop. Approximately, how long will the same machine take to sort $\boldsymbol{8\ times}$ as many ($\boldsymbol{8.000\ items}$)?

Q3) Identify the number of swaps required for sorting the following list in ascending order using selection sort and bubble sort and identify which is the better sorting technique with respect to the number of comparisons.

$\boldsymbol{List\ 1}$	63	42	21	9

Exercise II.3 (Insertion Sort)

Q1) **a)** Linear insertion sort has just correctly positioned $\boldsymbol{x[3]}$ in the following array $\boldsymbol{x}$:

$\boldsymbol{i}$	0	1	2	3	4	5
$\boldsymbol{x[i]}$	20	40	60	30	10	50

Show $\boldsymbol{x}$ after each of $\boldsymbol{x[4]}$ and $\boldsymbol{x[5]}$ is correctly positioned.

b) In what situation will linear insertion sort make the fewest interchanges?

Q2) Give the worst possible order of input for insertion sort with the following integers: $\boldsymbol{1, 2, 3, 4, 5, 6, 7}$. Assume that the result of sorting should be in ascending order.

Q3) The basic operation in the simple selection sort algorithm is to scan a list $x_1, x_2, \cdots, x_n$ to locate the smallest element and to position it at the beginning of the list. A variation of this approach is to locate both the smallest and the largest elements while scanning the list and to position them at the beginning and the end of the list, respectively. On the next scan this process is repeated for the sublist $x_2, x_1, \cdots, x_{n-1}$, and so on.

a) Using the array x, show x after the first two passes of this double-ended simple selection sort.

i	0	1	2	3	4	5
$x[i]$	30	50	70	10	40	60

b) Write a function to implement this double-ended simple selection sort.

c) Determine its computing time.

Exercise II.4 (Quicksort)

Q1) Which two techniques reduce the probability of Quicksort taking the worst execution time?

Q2) For the following array x.

i	0	1	2	3	4	5	6	7	8	9
$x[i]$	45	20	50	30	80	10	60	70	40	90

In Quicksort, show the contents of x after the function call : **Pivot = partition(x, 0, 10);** is executed , and give the value of the array index **Pivot**.

Q3) We make the call: **int Res = partition(Vec, 0, 6);**

for each of the 2 example arrays a given in the table below. Give the arrays after the call, and the value returned in **Res**. Use the partition method with the **LAST** item used as a pivot.

	0	1	2	3	4	5	6	**Res**
Original array **a** example 1	**13**	**7**	**12**	**8**	**6**	**15**	**10**	
Array example 1 after partition								

Original array **a** example 2	**17**	**11**	**12**	**16**	**3**	**8**	**9**	
Array example 2 after partition								

Q4) We make the call: **int Res = partition(Vec, 0, 6);**

for each of the 2 example arrays a given in the table below. Give the arrays after the call, and the value returned in **Res**. Use the partition method with **FIRST** item used as a pivot .

	0	1	2	3	4	5	6	**Res**
Original array **a** example 1	**13**	**7**	**12**	**8**	**6**	**15**	**10**	
Array example 1 after partition								

Original array **a** example 2	**17**	**11**	**12**	**16**	**3**	**8**	**9**	
Array example 2 after partition								

Q5) The core of the Quicksort algorithm is the partition algorithm that separates smaller items from larger items in an array. Here is an implementation of partition that partitions the values from index start to index stop inclusive.

```
public static int partition(String[ ] data, int start, int stop){
        String pivot = data[start];
        int low = start - 1;
        int high = stop + 1;
        while (low < high){
                do high-- ; while (data[high].compareTo(pivot) > 0);
                do low++; while (data[low].compareTo(pivot) < 0);
                if (low >= high)
                        break;
                String temp = data[low];
                data[low] = data[high];
                data[high] = temp;
         }
        return high;
    }
```

Suppose the variable **data** contained the following array:

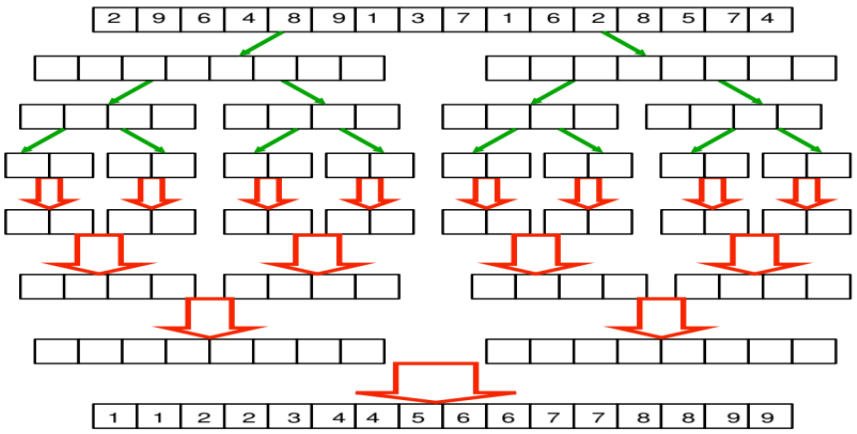
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
***data*[*i*]**	man	jay	ram	owl	eel	fly	kea	hen	ant	pig	tui	dab	cat	gnu	bat

Show the contents of **data** and **mid** (the return value) after the statement

int mid = partition(data, 0, 14);

Exercise II.5 (Merge Sort)

Q1) Complete the following diagram, showing the intermediate steps that merge-sort uses to sort the following array of integers from smallest to largest.



Q2) Show the array below after each call to the Merge (not Mergesort). Highlight the elements that are modified or "touched" by merge.

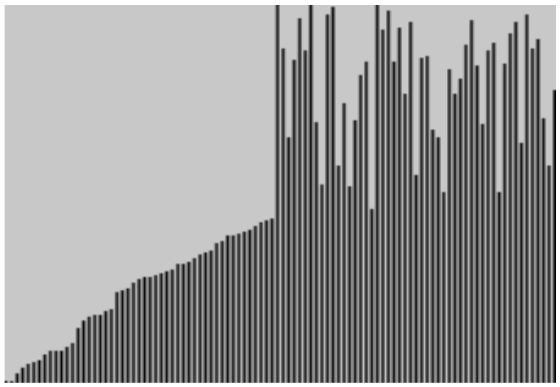
Index	0	1	2	3	4	5
Original array	15	11	12	13	17	10

Q3) Trace through Merge-Sort on the input (14, 11, 4, 13, 7, 6, 12, 5, 10, 8, 15, 9, 16, 3, 1, 2)

Exercise II.6 (Sorting Algorithms)

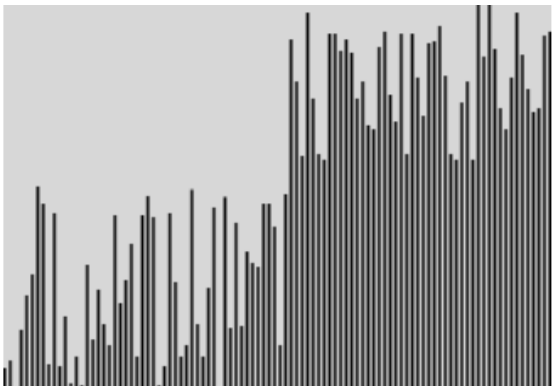
The lectures gave code for five common sorting algorithms: Bubble Sort, Selection Sort, • Insertion Sort, Quicksort and Merge Sort

Q1) Each of the examples below shows a different sorting algorithm caught part of the way through its operations on an array. Use your knowledge of the algorithms to decide which is being used in each case. Each algorithm appears once only. Write your answer below each picture.



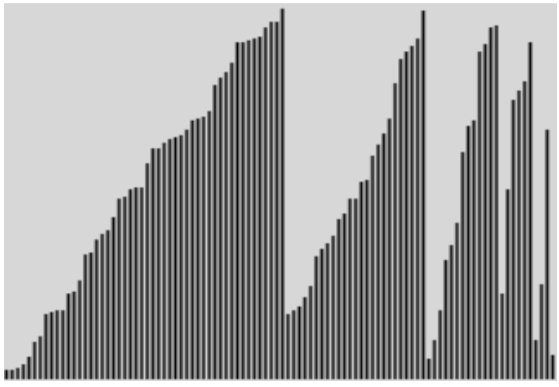
Algorithm: ……..

Justification: ……



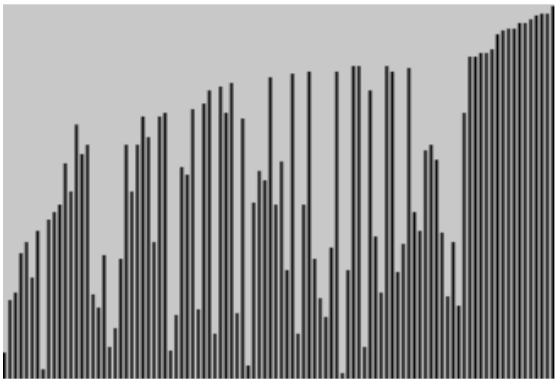
Algorithm: ……

Justification: ……



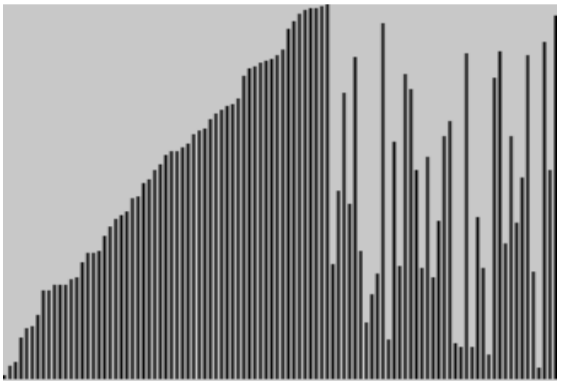
Algorithm: ……

Justification: ……



Algorithm: ……

Justification: ……



Algorithm: ……

Justification: …….

Q2) One algorithm performs much better when most of the items are already in order. Which one?

Exercise II.7 (Sorting Algorithm Steps)

For the following array $\boldsymbol{x = 0, 4, 2, 7, 6, 1, 3, 5.}$ Show the steps taken for each sort as we have learned in the lecture. Sort elements so that the result is ordered from **smallest to largest**.

Q1) Bubble Sort

Q2) Selection Sort.

Q3) Insertion Sort.

Q4) Quicksort (Assume that we always choose the last element as the pivot. Show the steps taken at each partitioning step.)

Q5) Merge Sort

Exercise II.8 (More Sorting Algorithm Steps)

For the following array $\boldsymbol{x}$.

$\boldsymbol{i}$	0	1	2	3	4
$\boldsymbol{x[i]}$	7	5	4	2	3

Q1) Demonstrate the changes made by each **swap** operation of bubble sort in ascending order when performed on the $\boldsymbol{x}$ array.

Q2) Demonstrate the minimum valued elements found during each iteration and the changes made by each **swap** operation of selection sort in ascending order when performed on the $\boldsymbol{x}$ array.

Q3) Demonstrate the changes made by each **swap** operation of insertion sort in ascending order when performed on the $\boldsymbol{x}$ array.

For the following array $\boldsymbol{y}$.

$\boldsymbol{i}$	0	1	2	3	4	5	6	7	8
$\boldsymbol{y[i]}$	8	5	7	2	6	1	4	9	3

Q4) Demonstrate how the array is sorted by showing which elements are selected as pivot elements, how sub-arrays are sorted against these pivot elements, and how the process is applied again to each sub-array when a Quicksort is performed on the $\boldsymbol{y}$ array.

Q5) Demonstrate how the array is first split apart, and then how it is merged when merge sort when performed on the $\boldsymbol{y}$ array:

Exercise II.9 (Time Complexity & Auxiliary Space & Sorting Algorithm)

Give the worst-case time complexity for each sorting algorithm, as well as the amount of additional memory required (in addition to that needed for input).

Sort	Worst Time	Auxiliary Space
Bubble	…	…
Selection	…	…
Insertion	…	…
Quick	…	…
Merge	…	…

Exercise II.10 (New Sorting Algorithm)

Professor Stankowski proposes the following algorithm for sorting an array $\boldsymbol{Vec}$ of $\boldsymbol{n}$ numbers:

- **i)** If there is only one number, return.
- **ii)** If there are two numbers, perform a single comparison to determine the order.
- **iii)** If there are more than two numbers, then first sort the top two-thirds of the elements recursively. Follow this by sorting the bottom two-thirds of the elements recursively and then sorting the top two-thirds of the elements again.

Q1) Write a recursive algorithm to implement the above strategy

Q2) What is the comparison complexity of Professor Stankowski's algorithm? Formulate a recurrence relation and solve the same to justify your answer.