Exercise Session 01

Exercise 1.
Consider the problem of finding the two smallest numbers in a nonempty sequence of numbers $\langle a_1, \ldots, a_n \rangle$ not necessarily sorted.

(a) Formalise the above as a computational problem (be careful to precisely define the input, the output, and their relationship).

Input: A non-empty array of numbers $(a_1, a_2, ..., a_n)$

Output: An array of two numbers $[a_{\pi 1}, a_{\pi 2}]$ that are the smallest elements of the original array and $a_{\pi 1} \leq a_{\pi 2}$

(b) Write the pseudocode of an algorithm that solves the above computational problem assuming the sequence is given as an array $A[1 \ldots n]$.

```
findTwoSmallest(A)
1  (if A.length < 2) return
2  smallest1 = A[1]
3  smallest2 = ∞
4  for index = 2 to A.length
5       if A[index] < smallest2
6            smallest2 = A[index]
7            if smallest2 < smallest1
8                 smallest1 with smallest2
9  return [smallest1, smallest2]
```

(c) Assume that line i of your pseudocode takes constant time c_i to execute. What is the worst case running time of your algorithm?

$$T(n) = c_1 + c_2 + c_3 + c_4 n + (c_5 + c_6 + c_7 + c_8) \cdot (n - 1) + c_9$$

Exercise 2.
Consider the following pseudocode for the function FIND-ELEMENT takes as input an array $A[1 \ldots n]$ and a number a.

```
FIND-ELEMENT(A, a)
1  for i = 1 to A.length
2       if A[i] = a
3            return i
4  return i
```

(a) Count the number of iterations of the for loop in the above pseudocode for the execution of FIND-ELEMENT$([1, 0, 5, 2, 4], 5)$ and report the value returned at the end of the call.
Because of the return i in the for loop the iteration only goes through the for loop 3 times when a i 5

(b) Is the above algorithm solving the element search problem presented in class? Justify your answer.
According to the defined element search problem in class the output should be the found array if it exist otherwise it should return 0. This algorithm returns i instead of 0 in the case that $a \notin A$

Exercise 3.
Write a pseudocode of an algorithm to reverse an array of numbers, i.e., the last element should become the first, the second last should become the second, etc. For example, the reverse of $[1, 2, 3, 4]$ is $[4, 3, 2, 1]$.

```
swap(A)
1  for index = 1 to ⌊A.length/2⌋
2      temp = A[index]
3      A[index] = A[A.length − index + 1]
4      A[A.length − index + 1] = temp
5  return A
```

(a) What is the worst-case running time of your algorithm?

$$T(n) = c_1(n/2) + (c_2 + c_3 + c_4) \cdot ((n/2) - 1) + c_5$$

(b) Try now to propose an algorithm that use only a constant amount of extra space. What is the worst-case running time of your algorithm?
Our algorithm already uses a constant amount of extra space in order to swap the elements of A, and the running time is the same as in (a). Moreover $O(n) = \Omega(n)$

Exercise 4.
For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds (1 microsecond $= 10^{-6}$ seconds).

	1 second	1 minute
$\lg n$		
$\sqrt{n}$		
$n \lg n$		
n^2		
n^3		
2^n		

In order to calculate largest size n in seconds: $f(n) \cdot 10^{-6} = 1$
In order to calculate the largest size n in 1 minutes: $f(n) \cdot 10^{-6} = 60$

	1 second	1 minute
$\lg n$	2^{10^6}	$2^{6 \cdot 10^7}$
$\sqrt{n}$	10^12	$3.6 \cdot 10^{15}$
$n \lg n$		
n^2		
n^3		
2^n		

★ **Exercise 5.**
Let A and B be two arrays of numbers sorted in non-decreasing order, respectively of length n and m. Write the pseudocode of an algorithm that checks whether the set of elements in A is equal to the set of elements in B, i.e., all elements of A are contained in B and vice versa. What is the worst-case running time of your algorithm?

Input: Two arrays of numbers $(a_1, a_2..., a_n)$ & $b_1, b_2..., b_m$ which are sorted in non-decreasing order.
Output: Return true if all elements of the set A is $= B$, otherwise return false.