# The National Higher School of Artificial Intelligence
# Data Structures and Algorithms
# Midterm Exam   30/11/2022
# Duration: 2h30 minutes

**Do not use any red or pink colour in your answers! OTHERWISE you will be PENALISED**

**You will get 1 bonus mark for clean, neat, and clear writing.**

## Exercise 1    (5.5 marks)

1. Giving a brief explanation, say how many different orderings of the four numbers {1, 2, 3, 4} there are?
2. By considering all those possible orderings, on your draft paper (brouillon) draw all possible binary search trees of size four with nodes labelled by the four numbers {1, 2, 3, 4}.
   After discarding any duplicate trees, draw on your answer paper all the different binary search trees. how many of size four are there? Label them in order: A, B, C, …, Z, A1, B1, C1, …, Z1, … (each BST you have taking one of these labels).
3. For each different tree, state its height, how many leaf nodes it has, and whether it is perfectly balanced. Use the BST labels you have used in the previous question to give the information requested here.

## Exercise 2       (4 marks)

The numbers [25, 12, 6, 17, 29, 18] need to be inserted one at a time, in that order, into an initially empty binary max-heap tree.

1. Draw the state of the max-heap tree after each number has been inserted.
2. Now draw the max-heap tree after each of the first two highest priority items have been removed each from the resulting Heap Tree.
3. Finally, draw the max-heap tree after each of the two removed items have been added back to the Heap Tree in the order they were removed.
4. To what extent does the order of the initial list of items affect the heap tree that is produced and the order in which the highest priority items are removed?

## Exercise 3      (4.5 marks)

A library has its books organized primarily according to 20 categories represented by the two-digit codes 01, 02, 03, … 20, and secondarily according to the first two letters of the first author's surname Aa, Ab, …, Az, Ba, …, Zz. Use Radix Sort to sort the set of books with keys: [`09 Ce, 09 Fa, 16 Mo, 16 Fa, 07 Ce, 13 Fa, 09 Mo, 07 Ba, 13 Ca`]. Show the state of the book list and what is being done at each stage.

## Exercise 4      (5 marks)

1. Suppose you have a hash table of size 19, the keys are words, and the hash map is defined as follows: each letter is assigned a number according to its position in the alphabet (i.e. a is assigned 0, b 1, …, z is assigned 25), and the primary hash function is "x modulo 19", where x is the number corresponding to the first letter of the word. Discuss if this hash function is very good or not?
2. Suppose instead you have a hash table of size 13 and the primary hash function is "x modulo 13", where x is the sum of the numbers corresponding to all the letters in the key word. Insert the following list of words into an initially empty hash table using linear probing:
   [`computer`, `science`, `in`, `birghanimm`, `dates`, `back`, `to`, `the`, `sixties`]
3. What is the load factor of the resulting table, and how many collisions occurred?
4. What is the effort (i.e. number of comparisons) involved in checking whether each of the following words are in the hash table: `teaching`, `research`, `admin`?

5. Show what the resulting hash table would look like if direct chaining had been used rather than linear probing.
6. Now what is the effort (i.e. number of comparisons, not including the processing of NULL pointers) involved in checking whether each of the following words are in the hash table: `teaching`, `research`, `admin`?

**Exercise 5 (6 marks)**

1. Write an efficient recursion-based procedure `isHeap(t)` that returns `true` if the binary tree `t` is a heap tree, and `false` if it is not. You can assume you can use the following standard primitive binary tree functions `isEmpty(t)`, `root(t)`, `left(t)` and `right(t)`, the first one returning a boolean value and the remaining three a pointer to the corresponding result. You can also use a function `complete(t)` that returns `true` if `t` is complete, and `false` if it is not. Make sure you consider all the cases.
2. What can be said about the overall complexity of your algorithm (i.e. including all the functions)?