Terms:

- Duration: 2 hours
- Documents, calculator and phone are not authorized
- The rate is indicative (±1)
- **One sheet per exercise**

Exercise 1 (9 points) – Lists

We consider a word as a list of characters on the alphabet {'a','b'}.

1) (3 points) Given a list of characters composed only of 'a' and 'b', write the iterative algorithm of the function *subfib(l: List<Char>): List<Char>* which returns the list of characters obtained by replacing in the initial list l each 'a' by "ab" and each 'b' by "a".
   Example : subfib("abaab") = "abaababa"
2) (3 points) Give the recursive algorithm of the function *subfib(l: List<Char>): List<Char>*.
3) (3 points) Give the recursive algorithm of the function *fibo(n: Integer): List<Char>* which returns the word obtained by applying n times the subfib function to the word "a" (a list only composed by the character' a ').

In addition to the functions defined during the lesson on the Abstract Type List, you can use the following function:

- *concat* (l_1,l_2: *List<Char>): List<Char>* which built a new list by concatenating the elements of the list l_1 with the elements of l_2 (Elements of l_1 followed by the elements of l_2 in the same list).

Exercise 2 (4 points) – Binary Trees

1) We want to write the recursive algorithm of the subroutine *traverse(a: BTree<Integer>, l: List<Integer>): Integer* which traverses the tree a according to the values in the list l containing only 0 or 1.
   - Starting from the root node of a, we read the elements of the list l. If the element is a 0, we traverse the left child, if it's a 1, we respectively traverse the right child. We continue the traversal following the same rules until the list is empty. This function must finally return the value of the current resulting node (for example if the list l contains 1, 0, 1 the function must return the value of the right child of the left child of the right child of the root node).
   - If the tree is empty, the function must return -1.
   - If the list is empty, the function must return the value of the root node of a.

Exercise 3 (7 points) – List of Lists

We consider a lists of points sorted in ascending order on the abscissa. We remind you that a point is characterized by a couple of real: its abscissa x and its ordinate y.

1) (4 points) Write the iterative algorithm of the subroutine *abscissaList(lp: List<Point>): L* which builds from a list of points a new list where each element is composed of an abscissa and the ordered list of y-coordinates of the points having the corresponding abscissa.
   Example:
   Consider the list of points LP = (1,2), (1,6), (3,7), (4,7), (4,6) (4,9).
   The resulting list is L = (1, (2,6)), (3, (7)), (4, (7,6,9))
2) (1 point) Give in C the complete declaration of the type of the list L.
3) (2 points) Give in C the complete declaration of the subroutine *abscissaList(lp: List<Point>): L.*