Modalités :

- Durée : 2 heures
- Aucun document autorisé, ni machine à calculer, ni téléphone
- Le barème est donné à titre indicatif (±1)
- **Une feuille par exercice**

Exercice 1 (9 points) – Listes.

On considère une liste d'ouvrages *LOuvrages* où chaque élément de la liste contient un *Livre* ; caractérisé par son code (un entier), son titre (une chaîne de caractères), et le nom de son auteur (une chaîne de caractères).

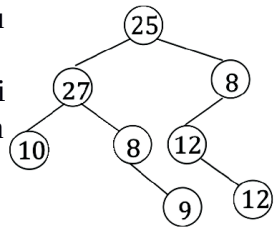
1) (3 points) Étant donné le nom d'un auteur et une liste d'ouvrages, écrire l'algorithme récursif du sous-programme **listerLivres** qui construit la liste des livres écrits par cet auteur. Pour chaque livre, on ne gardera dans cette liste que sa *Référence* composée de son code et de son titre. Vous pouvez utiliser (sans la définir) l'opération **egaleChaîne(ch1 : char*,ch2: char*): Booléen** qui donne vraie si les deux chaînes de caractères ch1 et ch2 sont égales, faux sinon.

2) (4 points) On souhaite construire une nouvelle liste *LAuteurs*, où chaque élément contient une valeur composée d'un nom d'auteur et la liste des Références de ses livres (composées du code et du titre). Écrire l'algorithme itératif du sous-programme **listerAuteurs** qui construit la liste *LAuteurs* à partir d'une liste de livres *LOuvrages*.

3) (2 points) Donner en langage C, la déclaration complète des types permettant de déclarer les listes *LOuvrages* et *LAuteurs* en utilisant une représentation chaînée.

Exercice 2 (7 points) – Cheminement d'Arbres binaires.

1) (4 points) On définit la longueur de cheminement interne d'un arbre binaire comme la somme des hauteurs des nœuds internes de l'arbre (nombre d'arêtes de la racine au nœud, les feuilles ne sont pas des nœuds internes).
Écrire l'algorithme récursif du sous-programme **cheminementInterne** qui prend en paramètre un arbre binaire d'entiers, et calcule cette longueur en utilisant uniquement les primitives du type abstrait Arbre Binaire.
Par exemple, la fonction renvoie 6 sur l'arbre ci-dessous.



2) (3 points) Écrire l'algorithme récursif du sous-programme **arbreVersListe** qui prend en entrée un arbre binaire d'entiers et un entier h et qui fournit en sortie la liste des valeurs des nœuds de hauteur h dans l'arbre.
Sur l'exemple de l'arbre ci-dessus, pour h=2, la liste résultat serait (10, 8, 12).

Exercice 3 (4 points) – Automates.

Construire des automates déterministes qui reconnaissent les langages suivants :
1) L'ensemble des mots sur l'alphabet {0, 1} qui contiennent au moins un 1. (2 points)
2) L'ensemble des mots sur l'alphabet {0, 1} qui contiennent exactement trois fois le symbole 1 (pas plus que 3 fois). (2 points)

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.

Consider a list of books *LBooks*. Each element of this list contains a *Book* which is characterized by its code (an integer), its title (a string of characters), and the name of its author (a string of characters).

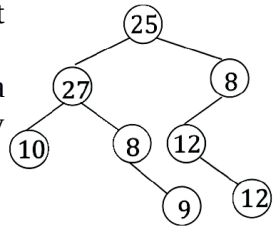
1) (4 points) Given the name of an author, write the recursive algorithm of the subroutine **listBooks** which builds the list of books written by this author. For each book, we will only keep BookReferences composed of book's code and title. You can use (without defining it) the operation **equalString(ch1: char *, ch2: char *): Boolean** that gives true if the two strings ch1 and ch2 are equal, false otherwise.

2) (4 points) We want to build a new list *LAuthors*, in which each element contains a value composed of an author name and the list of his books' references (composed of book's code and title). Write the iterative algorithm of the subroutine **listAuthors** that constructs the list *LAuthors* from *LBooks (*a list of books).

3) (1 point) Give in C language, the complete declaration of the types allowing to declare lists *LAuthors* and *LBooks* using a linked representation.

Exercise 2 (7 points) – Path in Binary Trees.

1) (4 points) We define the internal path length of a binary tree as the sum of the heights of the internal nodes of the considered binary tree (number of edges from the root to the node, the leaves are not internal nodes).
Write the recursive algorithm of the subroutine **internalPath** that takes a binary tree of integers as a parameter, and computes this length using only the primitives defined on the abstract data type *Binary Tree*.
For example, the function returns 6 on the tree below.



2) (3 points) Write the recursive algorithm of the subroutine **treeToList** which takes as input a binary tree of integers and an integer h and which outputs the list of values of nodes of height h in the tree.
On the example of the tree above, if h = 2, the result list would be (10, 8, 12).

Exercise 3 (4 points) – Automata.

Construct deterministic automata that recognize the following languages:
1) The set of words on the alphabet {0, 1} that contain at least a 1. (2 points)
2) The set of words on the alphabet {0, 1} that contain exactly three times the symbol 1 (no more than 3 times). (2 points)