Modalités :

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

Exercice 1 (9 points)

Une entreprise de fabrication de cartouches pour les imprimantes doit gérer la liste des commandes de ses clients. Chaque commande comporte l'identifiant du cartouche (un entier) commandée, un nombre d'exemplaires de cette pièce (un entier) et un identifiant du client concerné par la commande (une chaîne de caractères).

1) Etant donné l'identifiant d'un cartouche et une liste de commandes, écrire l'algorithme **récursif** du sous-programme qui construit la liste des commandes portant sur ce cartouche. Pour chaque commande de cette liste à construire, on ne gardera que le nombre d'exemplaires du cartouche et l'identifiant du client. **(3 points)**
2) On souhaite construire une liste optimisée des commandes LC, où chaque élément est composé de l'identifiant d'un cartouche et la liste des commandes (chaque commande ne contient plus que le nombre d'exemplaires et l'identifiant du client) portant sur ce cartouche. Ecrire l'algorithme **itératif** du sous-programme permettant de construire la liste LC. **(3 points)**
3) Donner en langage C, la déclaration complète des types permettant de déclarer la liste LC en utilisant une représentation chaînée. **(3 points)**

Exercice 2 (6 points)

On considère des arbres binaires booléens (composés uniquement de 0 et de 1). On dit qu'un arbre binaire booléen est OU-consistant si la valeur de chaque nœud qui n'est pas une feuille est le résultat de l'application du OU logique entre la valeur de son nœud fils gauche et la valeur de son nœud fils droit.

1. Etant donné un arbre binaire, écrire l'algorithme du sous-programme qui teste s'il est OU-consistant ou pas. **(3 points)**
2. Etant donné un arbre binaire, écrire l'algorithme récursif du sous-programme qui construit la liste contenant les valeurs de toutes ses feuilles. **(3 points)**

Exercice 3 (5 points)

Etant donné un arbre binaire d'entiers, écrire l'algorithme du sous-programme AjoutFeuilles qui ajoute des feuilles à cet arbre de la manière suivante : deux nouvelles feuilles sont ajoutées à chaque feuille de l'arbre donné. La valeur de chacune de ces nouvelles feuilles est la somme des valeurs des nœuds se trouvant sur le chemin menant de la racine aux anciennes feuilles.

Exemple :