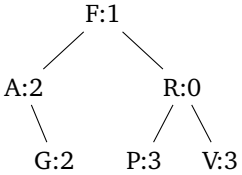
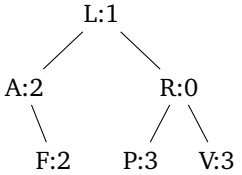
Contrôle continu

Le contrôle continu dure 1h30, sans aucun document autorisé ; le sujet comporte 3 exercices indépendants. Toute question non résolue peut être admise dans la suite. Les réponses doivent être correctement rédigées et justifiées. Le barème est sur 21 points.

Exercice 1 (sur 8 pts). *Du cours*

1. Pour chaque affirmation, indiquer si elle est **vraie ou fausse** en **justifiant**.
   - **i.** Dans un tas, le plus petit élément est une feuille.
   - **ii.** Dans un arbre binaire de recherche, le plus petit élément est une feuille.
   - **iii.** Un tableau trié par ordre décroissant est un tas.
2. Parmi les tableaux suivants, lesquels sont des tas : $T_1 = [7,3,6,1,2,4,5]$, $T_2 = [7,6,3,1,2,4,5]$, $T_3 = [7,6,5,1,2,3,4]$?
3. On considère les deux arbres binaires ci-dessous, dans lesquels les clés sont les lettres et les valeurs sont les entiers.





   - **i.** Lequel des deux est un arbre binaire de recherche ? *Donner explicitement le problème pour celui qui n'en est pas un.*
   - **ii.** On note D le dictionnaire représenté par l'arbre qui est un ABR, et on applique les opérations suivantes : $D_{[Q]} \leftarrow 2$; $D_{[A]} \leftarrow 1$; SUPPRIMER(D, L). Dessiner l'arbre obtenu après avoir appliqué les trois opérations.
4. Qu'est-ce qu'une famille universelle de fonctions de hachage ?

Exercice 2 (sur 6 pts). *Structure bizarre*

On définit un TAD « Liste bizarre » dynamique avec les opérations :

— $L \leftarrow$ NVLISTEBIZARRE() crée une liste bizarre vide ;
— AJOUTER(L, x) ajoute x à la liste bizarre L ;
— RÉDUIRE(L) supprime les $\lceil n/2 \rceil$ plus grands éléments de L, où $n =$ TAILLE(L).

On suppose avoir une réalisation de ce TAD où NVLISTEBIZARRE et AJOUTER ont une complexité $O(1)$, et RÉDUIRE un complexité $O(n)$.

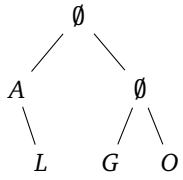
1. Justifier qu'en réalisant le TAD Liste bizarre à partir du TAD Liste du cours, les complexités de NVLISTEBIZARRE et AJOUTER sont celles annoncées. *On ne demande rien pour* RÉDUIRE.

On effectue une série de k opérations AJOUTER et RÉDUIRE. On veut montrer que le coût amorti par opération est constant. Soit c une constante telle que le coût de RÉDUIRE sur une liste de taille n est $\leq c \cdot n$. On définit le potentiel $\Phi_i = 2c \cdot n_i$ où n_i est le nombre d'éléments dans L après la $i^{\text{ème}}$ opération. *On rappelle que le coût amorti de la $i^{ème}$ opération est le vrai coût de l'opération plus la différence $\Phi_i - \Phi_{i-1}$.*

2. Montrer que le coût amorti d'une opération AJOUTER est borné par une constante.
3. Montrer que le coût amorti d'une opération RÉDUIRE est borné par une constante.
4. Que peut-on en déduire sur le coût total de k opérations AJOUTER et RÉDUIRE ?

Exercice 3 (sur 7 pts). *Arbre radix*

Un arbre radix est une réalisation du TAD dictionnaire quand les clés sont des mots binaires, sous forme d'arbre binaire. La *clé* d'un nœud est le chemin vers ce nœud, et définie de manière récursive : la clé de la racine est le mot vide ; si la clé d'un nœud est c, la clé de son enfant gauche (s'il existe) est $c{\cdot}0$ et celle de son enfant droit (s'il existe) est $c{\cdot}1$. Chaque nœud contient soit la valeur spéciale $\emptyset$ (vide), soit une (vraie) valeur v. Le dictionnaire est constitué des clés dont les valeurs sont $\neq \emptyset$.



Ex.: l'arbre radix ci-dessus représente le dictionnaire $\{0 : A, 01 : L, 10 : G, 11 : O\}$.

1. Dessiner l'arbre radix obtenu si on insère les couples $(1, I)$ et $(111, T)$ dans l'arbre ci-dessus.
2. **i.** Pourquoi peut-on supposer qu'aucune feuille n'est vide ?
ii. Exprimer la hauteur de l'arbre en fonction de la longueur des clés contenues dans le dictionnaire.
3. Proposer une réalisation en pseudo-code de l'opération RECHERCHER(D, c) du TAD dictionnaire et analyser sa complexité. *Rappel : si c n'est pas dans D, il faut renvoyer une erreur.*

On admet qu'on peut réaliser l'opération INSÉRER(D, c, v) en complexité $O(\ell)$ où ℓ est la longueur de la clé c.

4. Décrire un algorithme, basé sur un parcours d'arbre radix, qui prend en entrée un tableau de mots binaires et les affiche dans l'ordre lexicographique, en temps $O(N)$ où N est la somme des longueurs des mots du tableau. *Préciser le parcours d'arbre utilisé. L'ordre lexicographique sur les mots binaires est*

$$0, 00, 000, \ldots, 01, 010, 0100, \ldots, 011, 0110, \ldots, 1, 10, 100, \ldots, 11, 110, \ldots$$

Formellement, $w < w'$ s'il existe j tel que $w_{[0,j-1]} = w'_{[0,j-1]}$ et soit w est de taille j, soit $w_{[j]} = 0$ et $w'_{[j]} = 1$.