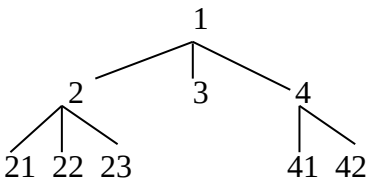
TP 2 : algorithme Minimax

Exercice 1 :
Un arbre quelconque peut être représenté par une liste dont le premier élément est la racine et les autres éléments les sous-arbres fils. Si un arbre est réduit à une feuille, il est représenté par cette feuille (supposée ne pas être une liste).

Exemple

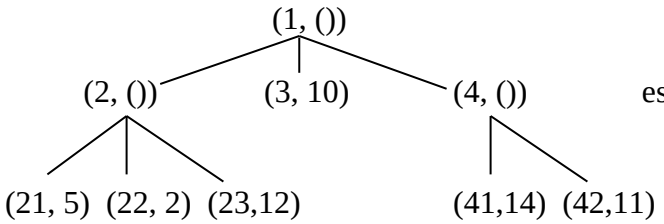


est représenté par : (1 (2 21 22 23) 3 (4 41 42))

Définir les fonctions suivantes :

- **arbre:cons** étant donnés un élément *r* et une liste d'arbres construit l'arbre de racine *r*.
- **arbre:feuille?** étant donné un arbre, détermine s'il s'agit d'une feuille.
- **arbre:lfils** étant donné un arbre, retourne la liste de ses sous-arbres fils.
- **arbre:racine** étant donné un arbre, retourne sa racine.
- **arbre:fils** étant donnés un arbre et un entier $k \geq 1$, retourne le k-ième sous-arbre fils.

Exercice 2 :
Un arbre de jeu peut être représenté en utilisant l'approche générale ci dessus, chaque nœud est représenté par sa valeur ou par () si le nœud n'est pas valué. Mais, pour une meilleure lisibilité de l'arbre, on représente un nœud par un couple (numéro, valeur), lorsqu'un nœud n'a pas de valeur, le second élément vaut ().



est représenté par : ((1) ((2) (21 5) (22 2) (23 12))
(3 10) ((4) (41 14) (42 11)))

1. Modifier, si nécessaire, les fonctions de l'exercice 1 pour cette nouvelle structure d'arbre. Attention, un arbre réduit à une feuille est maintenant représenté par une liste.
2. Définir la fonction **minimax** qui, étant donnés un arbre de jeu et la profondeur maximum de ses feuilles, implémente (en l'adaptant à la structure d'arbre donnée) l'algorithme Minimax vu en cours.
3. Tester vos fonctions sur les arbres donnés et vérifier la valeur Minimax de l'arbre. Le fichier *arbres.pdf* contient les arbres et le fichier *arbres.txt* leurs représentations.