TP6 partie 2 – Tuples, arbres et procédures

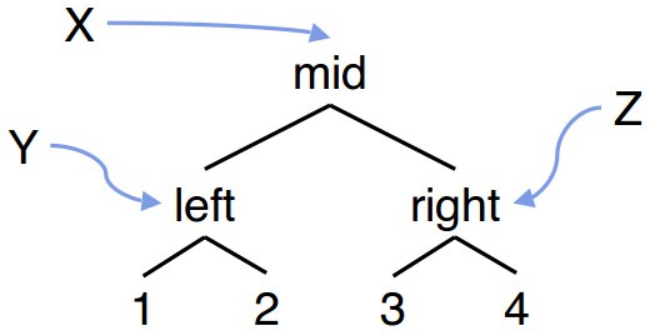
Dans ce TP, nous allons nous familiariser avec les **tuples** et les **arbres** qui font références aux chapitres 2.3 et 3.4.4.

Nous allons aussi continuer avec les procédures **procédure** que l'on peut trouver chapitre 2.3.4 et 2.4 du livre.

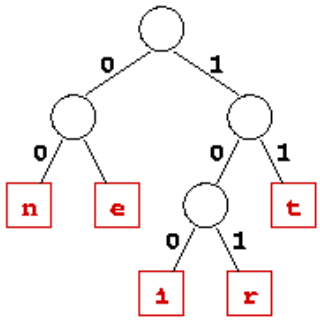
1. Les tuples et les arbres :

Question 1 : Créez un tuple ayant pour étiquette **« state »** et **combinant** 3 valeurs. Affichez ensuite son étiquette, sa largeur et le premier et troisième champs.

Question 2 : Créez un arbre composez de 3 tuples X, Y et Z



Question 3 : Dans cet exercice, nous allons utiliser un arbre binaire pour **décoder une suite de bits**. L'idée est la suivante: l'arbre contient dans chacune de ses feuilles un caractère. Voici la représentation de notre arbre où les feuilles sont les carrés. Implémenter cette arbre en Oz en utilisant les enregistrements.



Indice : <**arbre**> = leaf(<**lettre**>) | tree(0:<**arbre**> 1:<**arbre**>)

Question 4 : Le codage en lui-même est une suite de bits. Le codage d'une lettre représente le chemin dans l'arbre de la racine vers la feuille contenant cette lettre. Lorsqu'on va vers un fils gauche, on émet un 0, lorsqu'on va vers un fils droit, on émet un 1. Dans la figure, on a indiqué le bit correspondant à chaque branche. Par exemple, la lettre `i` *est codée comme la suite (1,0,0)*. Notez que les lettres ont des codes de longueurs différentes.
Écrivez une fonction **Decode** qui prend en arguments un arbre de codage (l'arbre que vous avez codé à la *question 4*) et une liste de bits (nombres 0 et 1). La fonction renvoie la liste des lettres correspondant à ce code.

Décodez la suite *[1 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1] !*

2. Les procédures, la suite :

Question 5 : On veut afficher une suite de texte à l'utilisateur. Le principe du programme est que l'utilisateur ne peut accéder aux textes suivant qu'après avoir confirmé avoir lu le texte courant.

Exemple : *« Bonjour, Sommes nous vendredi ? »*
« Tapez 1 pour confirmer que vous avez lu le message ci-dessus ! »
> 0
« Bonjour, Sommes nous vendredi ? »
« Tapez 1 pour confirmer que vous avez lu le message ci-dessus ! »
> 1
« Bientôt le weekend ! »

Écrivez le programme qui permet d'effectuer ce comportement.

Pour vous aider le programme ci dessous permet de récupérer les entrées claviers

```
declare
%Création d'une variable pour la récupération des entrées clavier
StdIn = {New class $ from Open.file Open.text end init(name:stdin)}
StringInput
Num = {NewCell 0}
in
{System.printInfo "Enter a string: "}
%On récupère l'entrée clavier que l'on stocke dans la variable StringInput
StringInput = {StdIn getS($)}
%On déclare que tant que la variable num ne contient pas la valeur 75000 on
répète les instructions ci-dessous
for until:@Num == 75000 do
{System.printInfo "Enter 75000: "}
Line = {StdIn getS($)}
in
Num := try {String.toInt Line} catch _ then 0 end
end
```