TP 2 – Structures de données

1 Listes

Une liste en Prolog est construite récursivement à l'aide d'un foncteur de la forme : `[_|_]`, la première place (premier underscore) indiquant le premier élément de la liste et la deuxième place la liste des éléments restants (c'est une liste). La liste vide est notée `[]`. En typant ces opérations, on aurait :

```
[] : -> liste
[_|_] : element liste -> liste
```

Cette écriture peut être allégée en Prolog. Ainsi la liste `[1|[2|[3|[]]]]` peut aussi s'écrire : `[1,2,3]`.

1. Définir un prédicat donnant la longueur d'une liste.
2. Définir un prédicat de la forme `estdans(Elem, Liste)` réussissant si l'élément est dans la liste. Donner des clauses définissant un prédicat permettant de trouver le n^e element d'une liste (et qui échoue si la liste est trop petite).
3. Écrire un prédicat permettant de concaténer deux listes, puis un autre pour renverser une liste.
4. Écrire un prédicat permettant de mettre à plat une liste dont les éléments sont soit simples, soit des listes non forcément à plat.

2 Chaînes de caractères

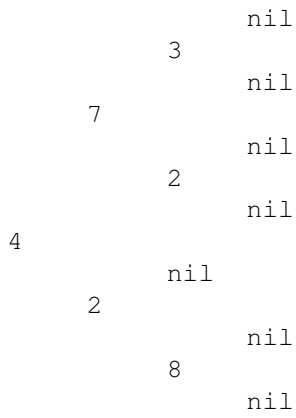
Dans les versions "standard" de Prolog (Prolog d'Edimbourg), les chaînes de caractères ne constituent pas une structure de données à part entière : les chaînes de caractères sont des listes d'entiers (les codes ASCII des caractères). Les constantes chaînes de caractères sont représentées en tapant la suite des caractères entre cotes ("). Ainsi, `write("salut")` produit sur la sortie standard : `[115,97,108,75,74]`. Le prédicat `name/2` transforme un atome en chaîne de caractères et inversement.

1. Écrire les clauses définissant un prédicat de nom `epelle/0` lisant un atome sur l'entrée standard et en écrivant ses lettres une par ligne.
2. Définir le prédicat `aconcat/3` réalisant la concaténation de deux atomes.
3. Définir le prédicat `substring/4` tel que `substring(Chaine, Debut, Nombre, Extrait)` réussit si `Extrait` est la sous-chaîne de `Chaine` commençant au caractère numéro `Debut` et comportant `Nombre` caractères. En déduire une définition Prolog de `subatom/4`, version des `substring` pour les atomes.
4. Définir un prédicat de nom `palin/1` testant si son argument (qui doit être un atome) est un palindrôme.

3 Arbres

Dans cette partie, nous considérons des arbres binaires d'entiers (par exemple) et les opérations associées en Prolog. Les arbres que nous considérons sont soit l'arbre vide (représenté, pour ce TP, par l'atome `nil`), soit un arbre de racine `X` et de sous-arbres gauche et droit respectivement `AG` et `AD` (représenté par le terme Prolog `t(AG, X, AD)`).

Question préliminaire : Quel terme Prolog correspond à l'arbre donné (horizontalement) ci-dessous ?



Définir, en Prolog les prédicats suivants :

`estarbin/1`	teste si son argument est un arbre binaire
`dans/2`	teste si un entier est contenu dans un arbre
`nn/2`	donne le nombre de noeuds de l'arbre
`hauteur/2`	donne la hauteur de l'arbre
`affiche/2`	affiche, horizontalement, l'arbre binaire donné en argument