TP 2 de Logique. Caml

On introduit les types suivants

```
type binconn = Et | Ou | Imp | Equiv;;

type formula = Var of string
             | Non of formula
             | Bin of binconn * formula * formula;;
```

La formule $(A \wedge B) \rightarrow ((\neg C) \vee (D \leftrightarrow E))$ sera par exemple représentée par

```
Bin(Imp,Bin(Et,Var "A",Var "B"),
Bin(Ou,Non(Var "C"),Bin(Equiv,Var "D",Var "E")))
```

On représentera une assignation de valeurs de vérité $v_1, v_2, \ldots, v_n$ à des proposisitons atomiques par la liste d'association `[(a1,v1);...;(an,vn)]`, c'est-à-dire une liste de couples de la forme (a, v) où a désigne la variable de proposition et v sa valeur de vérité, `true` ou `false`.

Exercice 1.— On définira les fonctions suivantes

- `hauteur f` donnant la hauteur d'une formule f.
- `atome f` donnant la liste des atomes propositionnels figurant dans la formule f.
- `assoc a l` qui a un element a d'une liste de couples associe un élément b tel (a, b) est dans la liste.
- `is_true f interpret` calculant la valeur de vérité de la formule `f` sous l'interprétation (ou l'assignation) `interpret`.
- `transforme f` qui transforme une formule f en une formule sémantiquement équivalente qui ne comporte que des négations sur les atomes propositionnels. On utlisera les lois de De Morgan.

Chercher les types des fonctions avant de les écrire. Tester les fonctions sur des exemples.

Exercice 2.— On définira les fonction suivantes

- `table varset` calculant l'ensemble de toutes les interprétations possibles d'un ensemble `varset` de variables.
- `is_valid f` décidant si la formule `f` est ou non une tautologie.