TPne - types produit et n-uplets

Exercice 1 - Construction de n-uplets

Assurez-vous que vous savez intuiter le résultat de l'évaluation de l'interprète OCaml sur chacune des expressions ci-dessous.

Contrôler vos réponses en utilisant l'onglet *Toplevel*.

```
# (1,false) ;;
- : int * bool = (1, false)

# (2., 1+4, 1=1, "salut") ;;
…

# (('a', 1), ('b', 2), ('c', 3)) ;;
…

# (1, 2, 3) ;;
…

# (1, (2, 3)) ;;
…

# ((1, 2), 3) ;;
…

# (1, 2, 3) = (2, 1, 3) ;;
…

# (1, (2, 3)) = (1, 2, 3) ;;
…
```

Définir une expression `quadruplet` dont la première composante est un booléen, la deuxième un couple d'entiers, la troisième une fonction dont le type est `float -> float` et la quatrième est une chaîne de caractères.

Exercice 2 - Filtrage sur les n-uplets

1) Étant donné 2 booléens, écrire en utilisant le filtrage (et en s'inspirant de la diapositive 14/17 du CTD2) les deux fonctions curryfiées suivantes :

- `imp` qui implémente la table de vérité de l'opérateur logique d'implication (A ⇒ B) ;
- `xor` qui implémente la table de vérité de l'opérateur logique « ou exclusif ».

On reprend l'exercice du TP1 sur les dates, en sachant qu'une date est maintenant représentée par un triplet d'entiers :
`(numéro du jour dans le mois, numéro du mois, année)`.

2) Écrire une version de la fonction `nbjour` prenant en argument un couple d'entiers, formé d'un numéro de mois et d'une année et retournant le nombre de jour de ce mois dans cette année. On

retournera `0` en cas de mois invalide.
Examinez le type de la fonction obtenue ; est-ce une fonction curryfiée ?

3) Écrire une fonction `valide` prenant en argument une date et retourne `true` si et seulement si la date est valide.

4) Écrire la fonction `lendemain` prenant en argument une date et retournant celle du lendemain. Une exception sera levée avec failwith en cas de date initiale invalide.

5) Écrire la fonction `veille` prenant en argument une date et retournant celle du la veille.
Une exception sera levée avec failwith en cas de date initiale invalide.

Exercice 3 - Filtrage sur les n-uplets

On manipule des temps représentés par un triplet `(heures, minutes, secondes)`.

1a) Écrire la fonction `sec2hms` de conversion d'un temps `ts` donné en secondes, en un temps exprimé en heures, minutes et secondes. Le type du résultat doit être un type produit comme cette fonction construit des triplets (mais il n'y a pas d'utilisation attendue du filtrage dans cette question).

1b) Définir l'expression `ex1` correspondant à la conversion de 3661 secondes en heures, minutes et secondes.

2a) Écrire la fonction `hms2sec` convertissant en temps donné en heures, minutes et secondes, en un temps exprimé en secondes.

2b) Écrire la fonction `sommeTemps` calculant la somme de deux temps `hms1` et `hms2`. Ces deux temps, ainsi que leur somme, seront donnés en heures, minutes et secondes.
(Suggestion : ne pas refaire un filtrage ici, mais utiliser les fonctions précédentes.)

2c) Définir l'expression `ex2` correspondant au temps exprimé en heures, minutes, secondes résultant de la somme de 1h 31min 7s et de 2h 29min 54s.