Programmation fonctionnelle avancée

Durée: 1h - Tout support interdit !

Nom:

Prénom:

Questions:

L'objectif du sujet proposé est de développer un interprète de logique descriptive utilisée dans le domaine du web sémantique vu en 2ème année.

1. La base de données considérée est donnée par la variable suivante, mais quel est son type ?

```
let db =
  [["ensisa"   ; "locatedAt"; "mulhouse"];
   ["mulhouse" ; "isA"      ; "Town"    ];
   ["ensisa"   ; "isA"      ; "School"  ]]
```

2. Quel sera le résultat affiché par le code suivant ?

```
db |> List.filter (fun [s;p;v]->p="isA" && v="School")
   |> List.map    (fun [s;p;v]->s)
   |> printfn "%A"
```

3. En vous inspirant du code précédent, définir une fonction `isA t` telle que le code suivant affiche `["mulhouse"]`.

```
db |> isA "Town" |> printfn "%A"
```

4. Sachant que la fonction `List.contains v l` permet de savoir si la valeur `v` est dans la liste `l`, comment définir un opérateur `(<&>)` retournant les éléments communs à deux listes (voir intersection de deux ensembles).

5. En reprenant la fonction `List.contains`, définir une fonction `restriction rel set` pour que le code suivant affiche `["ensisa"]`.

```
db |> restriction "locatedAt" ["mulhouse"] |> printfn "%A"
```

6. En considérant les définitions suivantes et les fonctions/opérateurs précédents, définir une fonction `eval exp db` pour évaluer une expression `exp`: `Prop` exprime une restriction et `And` une intersection. Ainsi, `eval e db` doit retourner `["mulhouse"]`.

```
type Exp =
  | IsA  of string
  | Prop of string*Exp
  | And  of Exp*Exp

let e = And (IsA "School", Prop ("locatedAt",IsA "Town"))
```

7. Définir une fonction générique `reduce exp f1 f2 f3` permettant de transformer `IsA` en `f1`, `Prop` en `f2`, et `And` en `f3`.

8. Ré-exprimer alors la fonction `eval exp db` en utilisant `reduce`.

9. Toujours en utilisant `reduce`, proposer une fonction permettant de sérialiser une expression. Par exemple, `serialize e` doit retourner le code suivant:

```
"School and locatedAt some Town"
```

10. Toujours en utilisant `reduce`, définir une fonction `map f exp` qui applique une fonction `f` à tous les valeurs (string) d'une expression. En particulier le code ci-dessous doit afficher:
`"Ecole and situéA some Ville"`.

```
let french term = match term with
  | "Town"      -> "Ville"
  | "locatedAt" -> "situéA"
  | _           -> term

e |> map french |> serialize |> printfn "%A"
```