Un TP monstrueux

groupe D2

12–12–2007

Modalités

- vous trouverez une liste exemple sur laquelle tester vos dernières fonction à l'adresse : `http://icps.u-strasbg.fr/~hoenen/monstre.ml`
- mailez moi le fichier contenant vos instructions Caml en tapant dans le terminal : `mail hoenen@icps -s [TP-algo]votre_nom < votre_fichier`

Ce type est un monstre

On veut créer un type monstre pour décrire des créatures fantastiques. Chaque monstre possède un nom, et 3 caractéristiques numériques entières non négatives : un nombre d'oeil, de bouche et de patte.

1. Définir le type monstre.
2. Écrire le constructeur du type monstre.
3. Écrire les accesseurs du type monstre.

Bébés monstres

Il existe deux moyens de faire des bébés monstres :

- *croisement* : si on les croise génétiquement, le croisement aura comme caractéristique la somme des caractéristiques des deux monstres dont il est issu.
- *reproduction* : si on laisse faire la nature, le bébé aura dans chaque caractéristique un nombre aléatoire comprit entre les caractéristiques de ses deux parents (bornes incluses).

Écrire deux fonctions *croisement* et *reproduction* qui à partir de deux monstres renvoient un nouveau monstre dont les caractéristiques suivent les règles ci-dessus. Si ces nouvelles caractéristiques sont les même que l'un des deux parents alors le nom du nouveau monstre sera identique à celui de son parent semblable. Si les nouvelles caractéristiques diffèrent, alors le nom du nouveau monstre sera la concaténation des noms des parents. On rappelle que l'opérateur de concaténation sur les `string` est `^` et que la fonction `Random.int 10` renvoie un entier aléatoire comprit entre 0 et 9 (donc `Random.int 0` est invalide).

Liste de monstres

1. Écrire une fonction qui prend une chaîne de caractère et une liste de monstres en entrée, et qui renvoie le nombre de fois ou un monstre portant ce nom est présent dans la liste.
2. Écrire une fonction qui prend en argument une liste de monstre, une des fonctions accesseur (oeil, bouche, patte), la valeur voulue pour cette caractéristique et qui renvoie la liste des monstres qui vérifient ce critère.
3. (bonus) Écrire une fonction qui supprime tous les doublons d'une liste de monstres (la liste résultat ne doit pas contenir plus qu'une fois un monstre). S'aider d'une fonction intermédiaire qui teste si un monstre est présent dans une liste.