

Javascript 2: Fonctions

jean-luc.falcone@unige.ch

Série 2 - Août 2019

1 Rosiers (2)

Pour cet exercices, il s'agit d'améliorer et d'étendre le code du dernier exercice. Vous pouvez partir de votre solution ou de la notre à choix.

1.1 Amélioration

Dans l'implémentation précédente, remplacez les boucles for ou while par des map ou des reduce.

1.2 Fonctions d'ordre supérieur

Ajouter les fonctions suivantes à vos arbres.

```
/* Applique la fonction 'f' sur les valeurs de l'arbre 'tree'.  
Celui-ci n'est pas modifié mais un autre arbre de même structure  
est retourné. */
```

```
function map( tree, f ) { ... }
```

```
/* Réduit l'arbre 'tree' par application répétée de la fonction  
binaire 'f'. L'arbre ne doit pas être modifié. */
```

```
function reduce( tree, f ) { ... }
```

Votre implémentation devra respecter les règles suivantes:

```
/* Pour tout arbre T, pour toute fonction unaire 'f' et 'g',  
et pour toute fonction binaire 'h' */  
size( map( T, f ) ) === size( T )
```

```

depth( map( T, f ) ) === depth( T )
map( T, (x)=>x ) === T;
toArray( map( T, f ) ) === toArray(T).map(f);
toArray( map( map( T, f ), g ) ) === toArray(T).map(f).map(g);

reduce( T, g ) === toArray(T).reduce(g)
reduce( map( T, (x) => 1 ), (x,y)=> x + y ) === size( T )

```

1.3 Utilisation

Implémentez les deux fonctions suivantes au moyen des fonctions que vous implémentées ci-dessus:

```

/* Retourne la somme des valeurs d'un arbres composé uniquement de nombres */
function sum( tree ) { ... }

/* Retourne la taille moyenne des valeurs d'un arbre composé
uniquement de chaînes de caractères */
function avgStr( tree ) { ... }

```