TD révisions 1

• Définir une fonction booléenne qui vérifie que tous les éléments d'une liste d'entiers sont multiples d'un entier passé en paramètre.

```
(multiples? '(5 15 35 20) 5) → #t
(multiples? '(5 15 36 20) 5) → #f
```

• Définir une fonction qui renvoie la somme des entiers pairs inférieurs ou égaux à un entier passé en paramètre.

```
(somme_pairs 11) → 30
```

• Définir une fonction qui, étant donnée une liste, construit la liste de ses nombres entiers associés à leurs carrés.

```
(n-carre '(a b 2 c 3.5 1)) → ((2 4) (1 1))
```

• Définir une fonction qui calcule le maximum d'une liste de nombres ainsi que le nombre d'occurrences de ce maximum (faire une version qui effectue les calculs en montant et une qui les fait en descendant).

```
(max-occ '(3 6 2 3 6 4 1 6)) → (6 3)
```

• Définir une fonction qui répète autant de fois un élément d'une liste que sa position dans la liste.

```
(repepete '(h e l l o )) → (h e e l l l l l l l o o o o o)
```

• Définir une fonction qui traite une liste en profondeur en remplaçant chacun des nombres par son opposé.

```
(oppose_en_profondeur '(2 c (4 -1 e) (f ((((-12)))) g h)))
          → (-2 c (-4 1 e) (f ((((12)))) g h))
```

• Définir une fonction qui calcule la moyenne des nombres d'un arbre de nombre en un seul parcours de l'arbre.

• Définir une fonction qui prend en paramètre un arbre binaire de nombres entiers et qui retourne un arbre où les valeurs sont des couples : la valeur du nœud et un booléen indiquant s'il est pair.

```
(construire-pairs '(2 (3 (4()())(5()()))(1()(6()()))))
→ ((2 #t)((3 #f)((4 #t)()())((5 #f)()()))((1 #f)()((6 #t)()())))
```