Feuille TD2.

TD2 : PF.

Exo1.

Q1.1 (a) Noeud 3 (Noeud 7 AV AV) (Noeud 1 AV AV)

(b) Noeud 2 (Noeud 3 (Noeud -1 AV AV) (Noeud 1 AV AV)) (Noeud -2 AV (Noeud 4 AV AV))

Q1.2

```
existe :: ArbreBin -> Int -> Bool

existe Vide _ = False
existe (Noeud a g d) v
    | v == a = True
    | otherwise = existe g a || existe d a
```

Q1.3

```
sommePos :: ArbreBin -> Int
sommePos Vide = 0
somme (Noeud v g d)
    | v < 0 = k
    | otherwise = v + k
  where
    k = somme g + somme d
```

Q1.4

```
ajoute :: Int -> ArbreBin -> ArbreBin
ajoute v Vide = Vide
ajoute v (Noeud v' g d) = Noeud (v+v') (ajoute v g) (ajoute v d)
```

Exo2

Q2.1

```
type Dictionnaire
  = Noeud (Int, Int) Dictionnaire Dictionnaire
  | Vide
```

Q2.2

```
decompte :: Dictionnaire -> Int -> Int
decompte Vide _ = 0
decompte (Noeud (a b) g d) v
    | a == v = b
    | a > v = decompte g v
    | otherwise = decompte d v
```

Q2.3

```
incrementer :: Dictionnaire -> Int -> Dictionnaire
incrementer Vide a = Noeud (a,1) Vide Vide
incrementer (Noeud (c,v) g d) a
        | c == a = Noeud (c, v+1) g d
        | c > a = Noeud (c,v) g (incrementer d a)
        | otherwise = Noeud (c,v) (incrementer g a) d
```

Q2.4.

```
compterAux1 :: [Int] -> Dictionnaire
compterAux1 [] = Vide
compterAux1 x:xs = incrementer (compterAux1 xes) x

compterAux2 :: Dictionnaire -> [(Int, Int)]
compterAux2 (Noeud (c,v) g d) = compterAux2 g ++ ((c,v) : compterAux2 d).

compter :: [Int] -> [(Int, Int)]
compter [] = []
compter l = compterAux2 (compterAux1 Vide l)
```