Exercice 1

Écrivez la fonction `powerset` qui prend en argument un ensemble, représenté par une liste, et qui retourne l'ensemble de tous les sous-ensembles de cet ensemble. Par exemple, pour l'ensemble {1,2}, cette fonction doit retourner {{},{1},{2},{1,2}}.

Attention : on vous demande d'écrire une fonction *polymorphe* qui fonctionne quel que soit le type des éléments de l'ensemble fourni en argument.

Exercice 2

On désire écrire une fonction qui, étant donnée une liste de pièces de monnaies, fournit toutes les sommes qu'il est possible d'obtenir au moyen de ces pièces.

Par exemple, si on dispose de deux pièces de 20 centimes et d'une pièce de 50 centimes, il est possible d'obtenir les sommes suivantes (en francs) : 0, 0.2, 0.2, 0.4, 0.5, 0.7, 0.7 et 0.9. Les doublons dans cette liste expriment le fait qu'il est parfois possible d'obtenir une même somme à partir de plusieurs combinaisons de pièces.

Écrivez une fonction qui, étant donnée une liste de nombres, retourne la liste de toutes les sommes qu'on peut obtenir en combinant ces nombres. Il n'est pas nécessaire que la liste retournée soit triée.

Indication : il existe une solution élégante utilisant le résultat de l'exercice précédent.

Exercice 3

Le type `int` de Scala est limité, au même titre que celui de Java, car les entiers sont représentés sur 32 bits. Le plus grand entier positif que l'on peut représenter avec ce type est $2^{31} - 1 = 2147483647$.

Pour contourner cette limitation (sans utiliser `scala.BigInt`), on décide de représenter les grands entiers positifs au moyen de listes d'entiers. Chaque élément de la liste est un chiffre du grand nombre, en base 10. Les chiffres les moins significatifs apparaissent en *tête* de liste. On suppose de plus que les nombres n'ont aucun zéro inutile. Ainsi, 0 est représenté par la liste vide `List()`, et 12340 est représenté par la liste `List(0,4,3,2,1)`.

(a) Écrivez une fonction qui incrémente de 1 un tel nombre. Indication :

```
def inc(n: List[int]): List[int] = {
  def inc1(n: List[int], carry: int): List[int]={ à compléter }
  inc1(n1, 0)
}
```

(b) Écrivez une fonction d'addition de tels nombres.