Homework 1 – Recursion and backtracking
ENSEA/FAME Computer Science

Due: Tuesday, February 25th 2020

The purpose of this homework is to design and implement an algorithm which decomposes a fraction as a sum of inverse squares. As an example, one can decompose:

$$\frac{7}{18} = \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{6^2}.$$

We only accept decompositions with *distinct* inverse squares. Thus, the following decomposition is forbidden:

$$\frac{1}{2} = \frac{1}{2^2} + \frac{1}{2^2}.$$

Question 1. Execute the following code:

```
>>> 0.2 + 0.1
>>> from fractions import Fraction
>>> Fraction(2, 10) + Fraction(1, 10)
```

Explain what happened.

Question 2. Write a function `def computeInverseSquaresSum(numbers)` which takes a list of positive integers `numbers` and returns a `Fraction` object equal to the sum of the inverse squares of its elements. Check that

```
>>> computeInverseSquaresSum([2, 3, 6])
Fraction(7, 18)
```

Question 3. Write a function `def findDecomposition(frac, upperBound)` whose arguments are a `Fraction` object `frac` and an integer `upperBound`, which returns either:

- a list $[a_1, a_2, \ldots, a_p]$ of positive integers $a_1 < \cdots < a_p <$ `upperBound` such that

$$\texttt{frac} = \frac{1}{a_1^2} + \cdots + \frac{1}{a_p^2},$$

- or `None` if there is no decomposition for `frac`

Do not forget to comment your code. *It is possible to use backtracking, however any working solution will be accepted.*

Question 4. Test with the following calls:

```
>>> findDecomposition(Fraction(7, 18), 7)
>>> findDecomposition(Fraction(1, 2), 40)
```

Check with the help of `computeInverseSquaresSum` that your result is indeed correct.