TP traitement des naturels et évaluation paresseuse

Préambule

Le module `paresse` proposé fournit 2 classes, `prendre` et `jeter`, qui mettent en œuvre les mêmes traitements que les fonctions `take` et `drop` de Haskell, de façon paresseuse.

I. Utilisation de mécanismes de l'évaluation paresseuse

1. Introduction

Dans cette première partie, le travail demandé consiste à utiliser des fonctions d'évaluation paresseuse pour effectuer des traitements sur les naturels. Il faudra utiliser les fonctions de base (`map`, `filter`), celles du module *functools* (`reduce`), celles du module *itertools* (`count`, `dropwhile`, `takewhile`) ainsi que celles du module *paresse* fourni (`jeter`, `prendre`).

2. Traitements à réaliser

1. Afficher la liste des naturels de 0 à 10 en utilisant `prendre`
2. Afficher la liste des naturels de 0 à 10 en utilisant `takewhile`
3. Afficher la liste des 20 premiers multiples de 3
4. Afficher la somme des nombres de 0 à 100
5. Afficher la liste des naturels de 100 à 200
6. Afficher la liste des naturels dont le carré est inférieur à 1024
7. Afficher la liste des naturels dont le carré est compris entre 1024 et 2048

II. Définition de fonctions d'ordre supérieur

Définir les fonctions `allmatch`, `anymatch` et `nonematch` :

1. `allmatch(a -> bool, iter[a]) -> bool` : renvoie vrai si et seulement si tous les éléments de l'itérable vérifient le prédicat ;
2. `anymatch(a -> bool, iter[a]) -> bool` : renvoie vrai si et seulement si au moins un élément de l'itérable vérifie le prédicat ;
3. `nonematch(a -> bool, iter[a]) -> bool` : renvoie vrai si et seulement si aucun élément de l'itérable ne vérifie le prédicat ;

III. Définition de fonctions d'évaluation paresseuse

Proposer des implantations des classes `Naturels`, `prendretantque` et `jetertantque`, remplaçants respectifs de `count`, `takewhile` et `dropwhile`.