Range series

Le contexte

- Le système sera initialisé avec un ensemble de ranges labélisés. Tous les ranges sont de type "closed-open" : [a..b)
- Un range est composé d'une borne inférieure (lower bound) et d'une borne supérieure (upper bound) et contient tous les éléments supérieur ou égal à 'a' et strictement inférieur à 'b'.
- L'ensemble des ranges est connus au départ et ne change pas au cours de l'exercice. La taille peut varier entre 1 & 10_000.
- Le but de l'exercice est d'implémenter la méthode : `marchingLabels(item: Item) : Label[]`
- La méthode `marchingLabels` peut être appellée plusieurs fois (entre 1 et 1_000_000 de fois)

Exercice :

Vous pouvez utiliser le language de votre choix : Java, Javascript, Typescript, C# ou tout autre language.

Le but est de réaliser le programme pour des ranges d'entiers et des libéllés sous forme de chaîne de caractères.

- Fournir un projet que l'on puisse réussir à executer facilement (en fournissant l'ensemble des fichiers nécéssaire à sont lancement : pom.xml, script de commandes, package.json ou autres)
- Modéliser les ranges au travers d'une variable nommée `ranges`
- Implémenter au moins une méthode ou fonction `marchingLabels(item)` qui doit retourner tous les libéllés des ranges qui contiennent l'`item` passé en paramètre.

Example:

- ranges : (A -> {0, 6}, B -> {5, 7})
  - matchingLabels(2) -> [A]
  - matchingLabels(5) -> [A, B]
  - matchingLabels(6) -> [B]
  - matchingLabels(8) -> []

Evaluation

Le test technique sera évalué selon le critères suivants:

- La qualité du code (simplicité, style, etc.)
- La qualité de la modélisation du problème
- La présence de différents cas de tests avec un jeu de données (de ranges et d'items)
- La compréhension de l'exercice et la complétude de la solution

- La performance de l'implémentation en terme de temps d'execution par rapport à nombre de ranges élevés