4. Partie Fonctionnelle (Scheme) [8 points][8% de votre note finale]

For this last part of the comprehensive assignment, we will ask you to implement the merging algorithm presented in part 3. This time the list of clusters will be given in a list of lists that is obtained by calling the provided `import` *function.*

Dans cette partie du projet intégrateur, nous vous demandons de réaliser à nouveau l'algorithme de fusion présenté à la partie 3. Cette fois la liste des groupes sera donnée dans une liste de listes obtenue en appelant la fonction `import` ci incluse.

```
> (import)
((65 1345 40.750304 -73.952031 65000001)
 (65 6017 40.760146 -73.957873 65000002)
 (65 17457 40.760213 -73.955471 65000003)
 (65 18582 40.750299 -73.952027 65000001)
 (65 20050 40.750365 -73.952127 65000001)
 (65 25351 40.760153 -73.955467 65000003)
 (65 34767 40.758621 -73.957704 65000004)
 (65 36487 40.758621 -73.957704 65000004)
…
```

The format is the same as before, that is:
`((PARTITION_ID POINT_ID X Y CLUSTER_ID) …)`
And we will work again with the same subset of partitions.

We ask you to create the function `mergeClusters` *that takes as input a list of clusters to be merged and output the list of clusters after merging.*

(`mergeClusters (import)`)

The resulting list must be saved in a text file as follows:

```
(define (saveList filename L)
 (call-with-output-file filename
  (lambda (out)
    (write L out))))

(saveList (mergeClusters (import)))
```

Make sure all your Scheme functions have a header describing what the function does, the input parameters and the output. Your function must adhere to the functional paradigm principles. In particular you must not use the functions terminating by ! (such as the `set!` function).

Le format de ces listes est le même qu'auparavant :
`((PARTITION_ID POINT_ID X Y CLUSTER_ID) …)`
Nous utiliserons ici le même sous-ensemble de partitions que précédemment.

Vous devez donc créer la fonction `mergeClusters` prenant en paramètres une liste de groupes à être fusionnés et qui retourne la liste des groupes fusionnés.

```
(mergeClusters (import))
```

La liste résultante doit être sauvergardée dans un fichier texte comme suit :

```
(define (saveList filename L)
 (call-with-output-file filename
  (lambda (out)
    (write L out))))

(saveList (mergeClusters (import)))
```

Remettre votre programme Scheme bien commenté. Inclure aussi les références aux resources utilisées. Votre solution doit se conformer au paradigme de programmation fonctionnelle. En particulier vous ne devez pas utiliser les fonctions se terminant par `!` (par exemple le `set!`).

Retour sur l'algorithme

The merge algorithm is relatively simple (at least in its non-efficient form). You add each cluster of the input list to the output list. But before you check if this cluster intersect with other clusters already in the list. If it is the case, you change the cluster ID of the intersecting clusters by the one of the cluster to be added. You then add the current cluster and repeat with the following one until there is no more clusters in the input list.

Here are therefore some important functions that you should define:

- A function that extracts a cluster from the main list; remove the first point of the list and all other points belonging to the same cluster.
- A function that computes the intersection between a cluster and a list of clusters. It returns the list of cluster IDs that have a point in common with the cluster.
- A function that changes the IDs of clusters by another one.

L'algorithme de fusion est relativement simple (du moins dans sa version non-efficace). Vous ajoutez chaque groupe de la liste d'entrée à la liste de sortie. Mais avant vous devez vérifier si ce groupe intersecte avec d'autres groupes déjà présents dans la liste. Si c'est le cas, vous changez le ID des groupes en intersection en leur donnant le ID du groupe courant. Vous ajoutez ensuite le groupe courant à la liste jusqu'à ce qu'il n'y ait plus de groupes à ajouter.

Voici donc quelques fonctions importantes que vous devriez définir :

- Une fonction qui extrait un groupe de la liste originales; pour ce faire retirer le premier de la liste et tous les autres points appartenant au même groupe.
- Une fonction qui calcule l'intersection entre un groupe et une liste de groupes. Elle retourne la liste des IDs de groupes ayant un point en commun avec ce groupe.
- Une fonction qui change les IDs de groupes pour un autre.