Pour ce dernier TP, vous allez implémenter un algorithme de jointure, *pour un maximum de 2 points*.

Vous avez le choix entre :

- simple nested loops join (plus facile à implémenter), *pour 1 point*.
- page-oriented nested loops join (un peu plus complexe), *pour 2 points.*

Vous avez normalement plein de briques déjà disponibles (sur la classe **HeapFile**) pour coder la logique nécessaire à la jointure.
Un conseil (mais pas une obligation) : rajouter une méthode **join** sur la classe **FileManager**.

A. Code : commande join

Rajoutez, dans votre application, la gestion de la commande **join.**
Cette commande calcule l'équijointure de deux relations spécifiées par leurs noms, pour deux colonnes spécifiées par leurs indices.

Le format de la commande est le suivant :
join nomRel1 nomRel2 indiceCol1 indiceCol2

La jointure à calculer est donc nomRel1><$_{\text{indiceCol1 = indiceCol2}}$ nomRel2.

Il n'est pas nécessaire de stocker le résultat sur disque (donc pas de buffer à réserver pour les résultats).

L'affichage doit suivre le même format que celui demandé pour la commande **select.**
Attention au schéma du résultat (n'oubliez pas des colonnes) !

B. Information : scénario de test jointures (fichiers R1.csv et S1.csv sur Moodle)

```
clean
create R 3 int string3 int
insert R 1 aab 2
insert R 2 abc 2
create S 2 int int
insert S 1 2
join R S 1 1                    (résultat attendu : 1 tuple : (1, aab, 2, 1, 2))
join R S 3 2                    (résultat attendu : 2 tuples : (1, aab, 2, 1, 2) et (2, abc, 2, 1, 2))
clean
create S 8 string2 int string4 float string5 int int int
insertall S S1.csv
create R 3 int string4 int
insertall R R1.csv
join R S 1 2                    (résultat attendu : 12 tuples)
join R S 2 3                    (résultat attendu : 7056 tuples)
join R S 3 6                    (résultat attendu : 320 tuples)
```