1 Manipulations de fichiers

On considère un fichier `liste_courses.txt` contenant les lignes suivantes :

```
lentilles
pommes
carottes
```

On considère le code suivant :

```
>>> with open('liste_courses.txt', 'r') as entree:
      liste = readlines(entree)
>>> liste
...
>>> liste.append('chocolat')
>>> liste.append('pain')
>>> with open('liste_courses.txt', 'a') as entree:
      entree.writelines(liste)
```

1. Compléter la valeur de `liste` sur les `...`, puis donner le contenu du fichier `liste_courses.txt`.
2. Et si on remplace le `open('liste_courses.txt', 'a')` par `liste_courses.txt', 'w')` ?

2 Ouvertures

Le but de l'exercice est d'identifier quel mode d'ouverture utiliser pour un fichier selon la spécification du programme.

On considère un jeu monojoueur qui manipule une grille représentée par une liste de listes et calcule un score en fin de partie gagnante. Les joueurs utilisent le surnom unique qui est associé à leur compte sur la plateforme de jeu. Les fonctionnalités proposées incluent :

a. Pour afficher des podiums : enregistrer dans un fichier le score d'un joueur en fin de partie gagnante, à raison d'un fichier par joueur et d'un score par ligne écrit à la fin du jeu ; le nom de ce fichier est `<surnom>.score`.

b. Pour pouvoir interrompre le jeu quand on le souhaite sans perdre le début de la partie : sauvegarder le contenu de la grille dans un fichier `partie.txt`. On ne peut sauvegarder qu'une seule partie en cours. On efface la partie sauvegardée précédemment si elle existe.

c. Pour pouvoir reprendre une partie interrompue : construire une liste de listes représentant la grille courante qui est décrite dans le fichier `partie.txt`.

3. Pour chacune des fonctionnalités précédentes, décrire l'instruction `open` adaptée.

3 Exercices de programmation

3.1 Fusion

On considère des fichiers tels que :

- chaque ligne contient un entier (pour simplifier)
- le fichier est trié par valeur croissante des entiers

On souhaite fusionner 2 tels fichiers, c'est à dire écrire un 3ème fichier qui réunit les entiers contenus dans les 2 fichiers, dans l'ordre croissant. On ne se préoccupera pas des éventuels doublons. Par exemple, si le 1er fichier contient les entiers `1 4 6 12` et le 2ème fichier contient les entiers `1 2 3 6 20 22` on trouvera dans le 3ème fichier les entiers `1 2 3 4 6 6 12 20 22`.

4. Proposer une solution qui lit le contenu des fichiers puis passe par un calcul sur une structure de données intermédiaire avant d'écrire le résultat dans un 3ème fichier.
5. Proposer une solution qui ne passe pas par une structure de données intermédiaire.

3.2 Précipitations

On considère un fichier de données `precipitations.csv` qui contient des dates, des mesures de précipitations (on mesure la pluie qui est tombée en millimètres) et de mesure de vent (en km par heure). Par exemple :

```
date;precipitations;vent
20231120;1.4;45.0
20231121;1.4;60.1
20231121;1.4;5.2
20231122;1.2;10.
20231123;0;2.
```

6. Proposer une solution qui permet :

- de calculer la moyenne des précipitations contenues dans le fichier
- de vérifier si oui ou non les dates sont triées par ordre croissant
- de vérifier si oui ou non les dates sont toutes différentes
- d'ajouter une ligne contenant une date et deux mesures (précipitations et vent) au fichier