EXERCICE 1: … ….

Un joueur de cartes reçoit 8 cartes dans un ordre non trié. Cet ensemble de cartes sera appelé la DONNE.
On donne le script de tri qu'il utilise pour ranger ses cartes:

```
Pour chaque carte de la donne :
Regarder à la fin de la main triée
Mémoriser la valeur de cette carte
Tant que la nouvelle carte va avant la carte de la main triée :
                        Avancer le regard d'une carte vers la gauche dans la main triée
Fin tant que
Insérer la nouvelle carte à gauche de la carte de la main triée qu'on vient de regarder
Fin pour chaque
```

1. S'agit-il d'un tri par INSERTION ou bien par SELECTION? Expliquer.

On utilise cet algorithme de tri pour ranger la liste suivante: **`L = ['c','a','b','e','d','g','f']`**

2. Utiliser le tableau pour tracer l'évolution de la liste au cours de l'exécution de cet algorithme.

	indice i du debut de la partie non triée	indice de la fin de la liste	liste après itération	nombre de comparaisons effectuées
avant l'itération 1				
pour l'itération 1				
pour l'itération 2				
pour l'itération 3				
…				

3. Combien d'étapes sont nécessaires pour ranger les éléments de cette liste?

4. Combien de comparaisons entre les éléments de liste sont effectuées au total?

5. Combien de comparaisons y aurait-il si la liste faisait la même taille, mais avec des éléments placés en sens inverse? **`L = ['g','f','e','d','c','b','a']`**

EXERCICE 2: TRI PAR SÉLECTION

Le joueur utilise maintenant une autre technique de rangement, dont voici l'algorithme:

```
Pour chaque carte de la donne :
    Regarder à la fin de la main triée
    Mémoriser la position i de cette carte
    Rechercher la carte la plus petite dans la partie non triée (entre i et la fin de
la donne). Lire la position j.
    Permuter les cartes i et j.
Fin pour chaque
```

1. Ecrire le script python correspondant. On suppose qu'il existe une fonction recherche_du_min, qui prend en paramètre un **`tableau T, ainsi que 2 bornes a et b`** de recherche dans ce tableau, et qui retourne l'indice du minimum.

```
1 . …
2 . …
3 . …
4 . …
5 . …
6 . …
7 . …
8 . …
9 . …
```

On utilise cet algorithme de tri pour ranger la liste suivante: **`L = ['c','a','b','e','d','g','f']`**

2. Utiliser le tableau pour tracer l'évolution de la liste au cours de l'exécution de cet algorithme.

	indice i du debut de la partie non triée	indice de la fin de la liste	liste après itération	nombre de comparaisons effectuées
avant l'itération 1				
pour l'itération 1				
pour l'itération 2				
pour l'itération 3				
…				

3. Combien d'étapes sont nécessaires pour ranger les éléments de cette liste?

4. Combien de comparaisons entre les éléments de liste sont effectuées au total?

5. Combien de comparaisons y aurait-il si la liste faisait la même taille, mais avec des éléments placés en sens inverse? **`L = ['g','f','e','d','c','b','a']`**

6. Y-a-t-il un algorithme un peu plus efficace parmi les 2 étudiés (insertion et sélection)?