[insertion](http://127.0.0.1:8000/nsi_tris/tri_insertion/)**Exercice sur le tri par insertion**

Cet exercice a pour but de vous permettre de retrouver l’**algorithme du tri par insertion**.

Voici un rappel du **principe** de ce tri. On cherche à trier la **suite de cartes** suivantes :

12

10

8

6

4

11

9

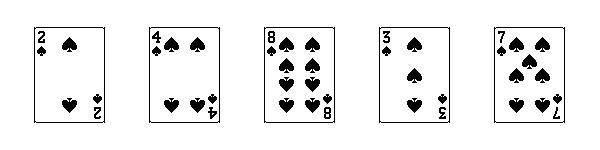
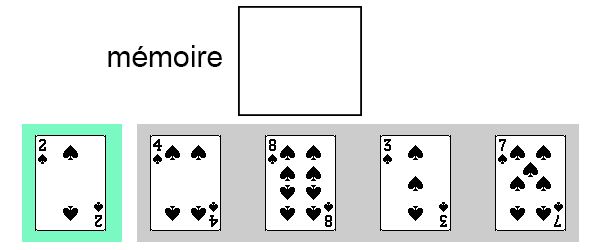
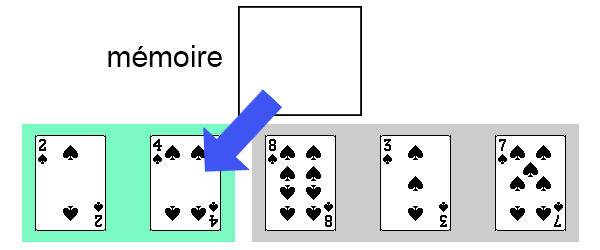
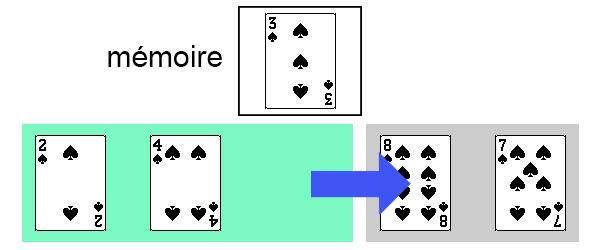
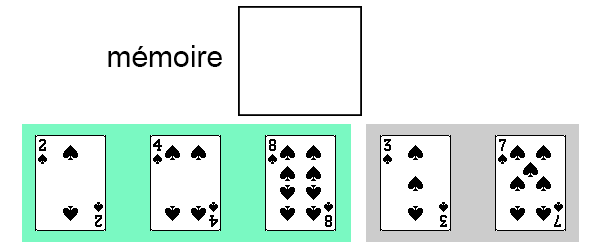
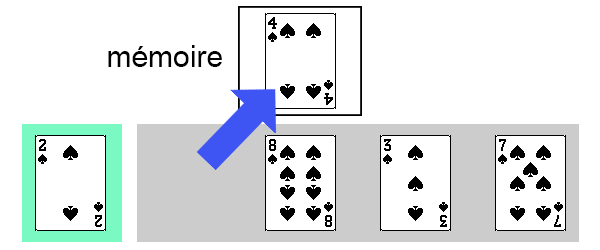
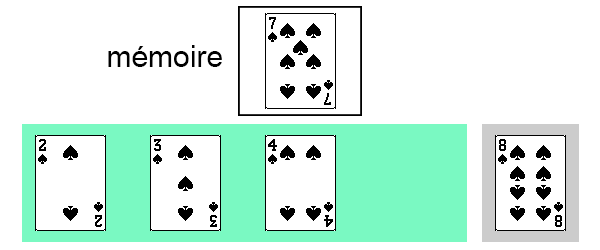
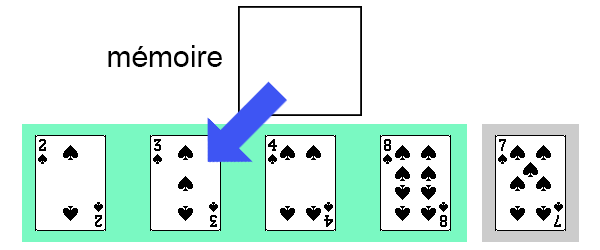
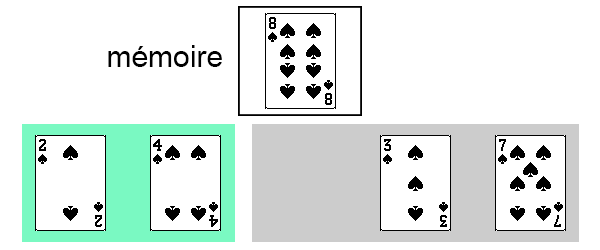
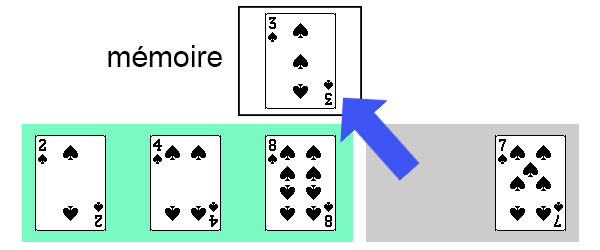
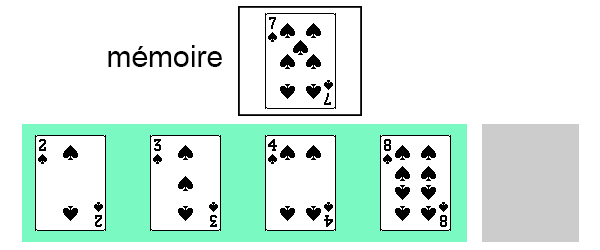
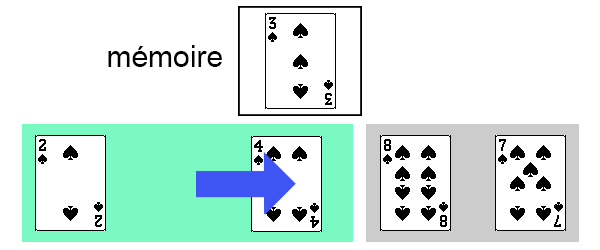
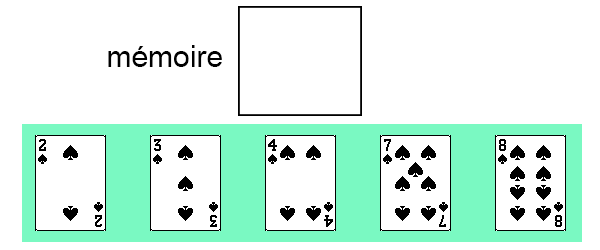
7

5

3

2

1



**Exercice 1 :**

**Complétez le principe du tri par insertion d’un tableau d’éléments.**

On considère que le **tableau** est composé d’une **partie triée** et d’une **partie non-triée**. La **partie triée** est initialement composée du **premier** élément du tableau.

* Pour trier un **tableau** en utilisant le **tri par insertion**, on commence par traiter le deuxième élément du tableau (on considère le premier élément comme **déjà trié**).
* Cet élément est considéré comme notre "clé" et est placé en mémoire .
* Maintenant, il faut comparer cette clé avec chaque élément précédent dans le tableau, en commençant par l'élément immédiatement à gauche et en continuant jusqu'à ce que l’on trouve un élément qui n'est pas plus grand que la clé, ou jusqu'à ce que l’on atteigne le début du tableau.
* Pendant ce processus, chaque élément qui est plus grand que la clé doit être déplacé vers la droite .
* Enfin, une fois que l’on a trouvé la position correcte pour la clé, on la récupère depuis la mémoire et on l’insère à sa bonne position.
* On répète cette procédure pour chaque élément du tableau, en allant de gauche à droite . À la fin, le tableau sera trié par ordre croissant .

**Exercice 2** :

**Répondre aux questions suivantes.**On notera **n** la longueur du **tableau** traité.

1. L’algorithme du tri par insertion parcourt tous les éléments du tableau un par un, on utilisera pour cela une **boucle POUR**. À partir de quel **indice** commencera cette boucle, et à quel **indice** s’arrêtera-t-elle ? Complétez l’instruction :

POUR **i** ALLANT DE 1 à n - 1 :

(corps de la boucle POUR)

1. La « clé » est l’élément actuellement considéré pour l'insertion dans la **partie déjà triée** du tableau. Il s’agit de l’élément d’indice **i** du tableau. Quelle instruction utilisera-t-on pour stocker cet élément en mémoire ?

clé = tableau[i]

1. On aura ensuite besoin d’une autre boucle pour comparer la « clé » avec chaque élément dans la **partie déjà triée** du tableau. Cette boucle doit aller **à l'envers**, en commençant par l'élément **immédiatement avant** la « clé » (d’indice **i**) et en continuant jusqu'à ce qu'elle trouve un élément qui n'est **pas plus grand** que la « clé », ou jusqu'à ce qu'elle atteigne le **début du tableau**. Pendant cette boucle, chaque élément qui est **plus grand** que la « clé » sera **déplacé vers la droite**. On utilisera pour cela une **boucle TANT QUE**. Complétez :

j = i - 1

TANT QUE j >= 0 ET tableau[j] > cle :

tableau[j + 1] = tableau[j] (déplacement de l’élément d’indice **j** à sa droite)

j = j – 1 (on passe à l’élément à gauche de **j**)

1. Enfin, une fois que la **boucle interne** est terminée, on a trouvé la bonne place pour la « clé ». Elle doit ainsi être inséré à la position **juste après** l'élément qui n'était **pas plus grand que la « clé »** (ou au début du tableau si aucun tel élément n'a été trouvé). Quelle instruction permet de replacer la « clé » à la bonne position ?

tableau[j + 1] = cle

**Exercice 3** : En vous aidant des deux exercices précédents, **écrire l’algorithme du tri par insertion**.