Nom Prénom : ..... Heure de passage : .....

BTS SIO
Sous-épreuve E22
Algorithmique appliquée
Contrôle en Cours de Formation

Déroulement de l'épreuve

Cette épreuve de Contrôle en cours de Formation (CCF) se déroule en trois étapes :

- **Écrit (30 minutes)**

  Vous devez traiter l'étape 1 du sujet. Pour cette partie, l'ordinateur est interdit mais la calculatrice est autorisée.

  Vous inscrirez vos réponses dans le document réponse à la fin du sujet.

  Les algorithmes à écrire peuvent être rédigés en **langage naturel** ou en Python mais ni en C# ni en VB.Net.

  À la fin de l'étape 1, votre document réponse doit être remis à la personne surveillant l'épreuve. Vous garderez le sujet.

- **Machine (30 minutes)**

  Vous devez traiter l'étape 2 du sujet à l'aide d'un ordinateur. Le langage utilisé est celui travaillé dans l'année, à savoir Python. Vous sauvegarderez votre travail sur la clé USB fournie.

  La durée totale pour effectuer les deux premières étapes est exactement d'une heure.

- **Oral (20 minutes au maximum)**
  Cette partie se déroule en deux temps. Tout d'abord, vous disposez de 10 minutes pour présenter votre travail de l'étape 2 puis, au cours des 10 minutes suivantes, un entretien permet de préciser votre démarche.

À la fin de l'épreuve le sujet devra être rendu à l'examinateur.

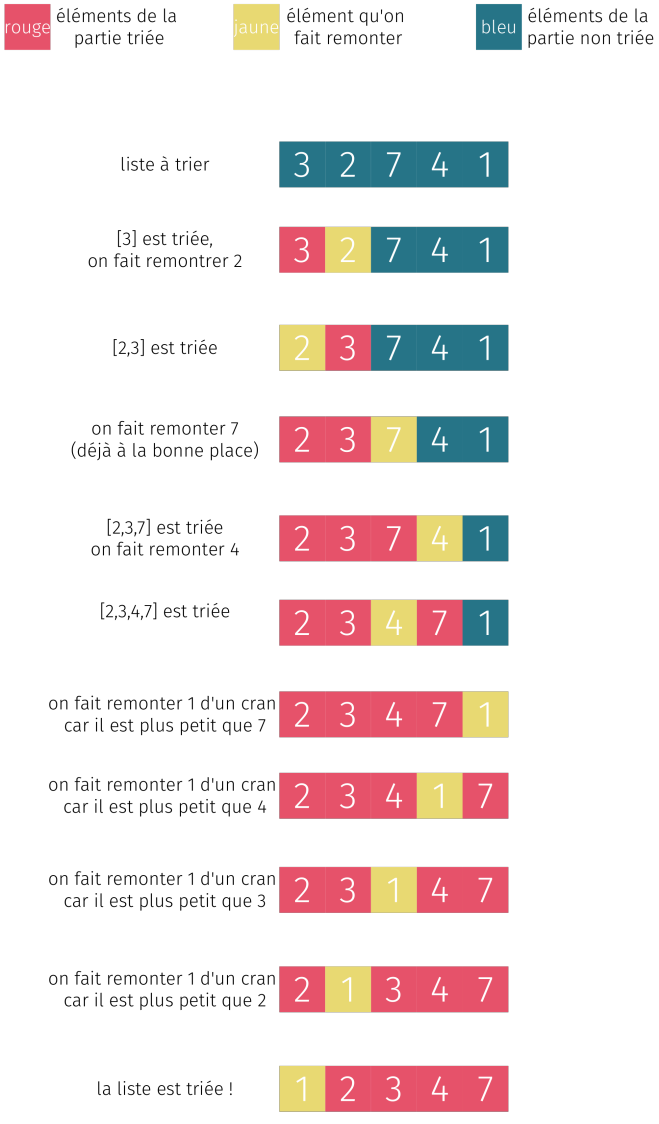
Tri par insertion

On dispose d'une liste d'entiers `lst` que l'on aimerait trier dans l'ordre croissant.
Nous allons utiliser l'algorithme de tri par insertion qui consiste à partager la liste en deux parties : la partie de gauche, qui devra à tout moment être triée et la partie de droite, qui comporte les autres éléments, non triés.

Au début de l'algorithme, la partie triée est composée seulement du premier élément `lst[0]` et la partie non triée du reste, puis

- on prend le premier élément de la partie de droite et on le fait « remonter » dans la partie de gauche en l'échangeant au fur et à mesure, jusqu'à ce qu'il soit à sa place ;
- la partie de gauche est triée et comporte un élément de plus, celle de droite comporte un élément de moins ;
- on recommence ceci jusqu'à ce que la partie de droite soit vide.

Voici un exemple :



Question 1

> Applique cet algorithme étape par étape en faisant comme à l'exemple précédent avec `lst = [4, 2, 8, 0, 3]`

On a commencé à écrire l'algorithme de la fonction `insere` qui

- en entrée prend une liste `lst` et un entier `i` : les éléments de `lst[0]` à `lst[i - 1]` sont triés et on veut faire « remonter » `lst[i]` parmi eux ;
- ne renvoie rien mais fait remonter `lst[i]` au bon endroit (il échange `lst[i]` avec `lst[i - 1]` tant que celui-ci existe et est plus grand que `lst[i]`);

Par exemple si `lst = [4, 6, 7, 5, 1]` `insere(lst ,3)` doit donner une valeur de `lst` de `[4, 5, 6, 7, 1]`.

Question 2

> Complète sur ta copie l'algorithme de la fonction `insere` .
>
> ```
> fonction insere(lst, i)
>     variables
>         temp : entier
>     tant que i < 0 et ... repeter
>         temp ← lst[i - 1]
>         lst[i - 1] ← ...
>         lst[i] ← ...
>         i ← ...
>     fin tant que
> ```

On a commencé à écrire la fonction `tri_insertion` qui

- en entrée prend la liste `lst` à trier ;
- ne renvoie rien mais trie `lst` en appliquant le principe de tri par insertion exposé plus haut.

Question 3

> Complète sur ta copie l'algorithme de la fonction.
>
> ```
> fonction tri_insertion(lst)
>     variables
>         i : entiers
>     pour i allant de ... à ... repeter
>         ...
>     fin pour
> ```

Étape 2

Question 4

> Ouvrir le fichier `tri_insertion.py` et coder les fonctions manquantes.

Question 5

> En s'inspirant de ce qui a été fait, créer une fonction `tri_insertion_decroissant` qui trie les listes dans l'ordre décroissant.