

# LIFAP1 – Partie A - Algorithmique

## Contrôle Continu Terminal (Durée totale : 2h)

Lundi 14 décembre 2020

**Recommandations :** Les documents, calculatrice, téléphone portable sont interdits. La qualité de l'écriture et de la présentation seront prises en compte dans la note finale. Vous veillerez à **respecter** les notations et les règles d'écriture des algorithmes vues en cours et en TD. Un soin tout particulier devra être apporté à l'écriture des entêtes des différents sous-programmes.

NOM :

PRENOM :

Numéro Etudiant :

### Exercice 1 : Tri par insertion ou tri du joueur de cartes

On souhaite écrire un programme permettant de trier un tableau d'entiers. Chacun des éléments de ce tableau sera inséré dans un nouveau tableau résultat de manière à ce que les éléments restent triés tout au long de l'exécution de l'algorithme. La taille maximale des tableaux notée `MAX` est une constante fixée à 50, et le nombre de cases remplies à chaque étape noté `tailleT` sera passé **en paramètre de tous les sous-programmes**.

Exemple : on dispose du tableau suivant

1	4	6	8	12					
---	---	---	---	----	--	--	--	--	--

On souhaite ajouter la valeur 11 dans le tableau,

1	4	6	8	11	12				
---	---	---	---	----	----	--	--	--	--

puis la valeur 3.


1	3	4	6	8	11	12			
---	---	---	---	---	----	----	--	--	--

1. Ecrire l'algorithme d'une **fonction** `trouve_pos` qui à partir d'un tableau d'entiers `T` (dont les `tailleT` premières cases seront remplies) et d'une valeur `val` retournera l'**indice** de la case dans laquelle `val` doit être insérée dans le tableau `T` afin de respecter l'ordre croissant. Dans l'exemple ci-dessus, pour la valeur 11 on aurait l'indice 4 et pour la valeur 3 on obtiendrait l'indice 1 comme résultats.

Pour insérer une valeur dans un tableau à un indice donné, il faut procéder à un décalage de toutes les valeurs qui la suivent dans le tableau.

Exemple : pour insérer 3 dans ce tableau à la position 1

1	4	6	8	11	12				
---	---	---	---	----	----	--	--	--	--



Il faut décaler toutes les valeurs à partir de la position 1 puis insérer la valeur à l'indice 1. Attention, `tailleT` aura été incrémenté de 1 au final !!

1	3	4	6	8	11	12			
---	---	---	---	---	----	----	--	--	--

2. Ecrire l'algorithme d'une **procédure** `insere_valeur` qui à partir du tableau `T` contenant `tailleT` valeurs, insère la valeur `val` à l'indice `ind` donnés en paramètres en suivant l'algorithme décrit précédemment.

3. En utilisant les sous-programmes écrits en 1 et 2, écrire l'algorithme d'une procédure `tri_insertion` qui construit un tableau `T2` trié en insérant successivement chacun des éléments du tableau `T1`. Les tableaux `T1`, `T2` et la `tailleT` seront passés en paramètres.

## Exercice 2 : Distance de Hamming

La distance de Hamming entre deux mots (chaines de caractères) **de même longueur** est égale au nombre de lettres, à la même position, qui diffèrent.

Par exemple la distance de Hamming entre "rose" et "ruse" est de 1, entre "pomme" et "poire" est de 2.

- 1- Ecrire l'algorithme d'un sous-programme `met_a_la_meme_longueur` qui à partir de deux chaines de caractères `ch1` et `ch2` passées en paramètres, tronque la chaine la plus longue à la longueur de la plus courte.

Exemple si `ch1 = "cestbientotlafin"` et `ch2 = "boncourage"`, le sous-programme devra transformer `ch1` en `"cestbiento"`. On pourra utiliser la fonction `longueur` qui retourne la longueur d'une chaine.

- 2- Ecrire l'algorithme d'une **fonction** `distance_hamming` qui calcule la distance de Hamming entre deux chaînes de caractères de même longueur `ch1` et `ch2` passées en paramètres.

- 3- Ecrire l'algorithme du programme principal qui demande à l'utilisateur 2 chaînes de caractères, et, après les avoir mises à la même longueur, calcule et affiche la distance de Hamming entre ces deux chaînes. Vous utiliserez les sous-programmes écrits dans les questions précédentes.