NOM et Prénom :

Mardi 23 septembre 2025

Term Spé N.S.I.

Devoir (1h50)

(Calculatrice non autorisée)

Exercice 1 – Tri par sélection (4 points)

1°) Effectuer à la main un tri par sélection de la liste **tab** = [7, 3, 2, 4, 1, 8, 6, 9, 2] :

Indiquer l'évolution du contenu de la liste jusqu'à son état final dans ce tableau :

Etat initial du tableau : **tab** = [7, 3, 2, 4, 1, 8, 6, 9, 2]

[7, 3, 2, 4, 1, 8, 6, 9, 2]
[1, 3, 2, 4, 7, 8, 6, 9, 2]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]
[..., ..., ..., ..., ..., ..., ..., ..., ...]

Etat final du tableau : **tab** = [..., ..., ..., ..., ..., ..., ..., ..., ...]

2°) Compléter la fonction `tri_selection(tab)` ci-dessous :

```
def tri_selection(tab) :  
    """  
    Tri par sélection  
    Renvoie la liste tab triée dans l'ordre croissant  
    en utilisant l'algorithme du cours de 1ère NSI  
    ATTENTION : la liste tab sera modifiée  
    """  
    n = . . . . .  
    for i in range(. . . . .) :  
        indice_min = . . . . .  
        valeur_min = tab[. . . . .]  
        for indice in range(. . . . ., . . . . .) :  
            valeur = tab[. . . . .]  
            if . . . . . < . . . . . :  
                indice_min = . . . . .  
                valeur_min = . . . . .  
        tab[. . . . .] = tab[. . . . .]  
        tab[. . . . .] = . . . . .  
    return tab
```

Exercice 2 – Tri par insertion (4 points)

1°) Compléter la fonction `tri_insertion(tab)` ci-dessous :

```
def tri_insertion(tab) :  
    """  
    Tri par insertion  
    Renvoie la liste tab triée dans l'ordre croissant  
    en utilisant l'algorithme du cours de 1ère NSI  
    ATTENTION : la liste tab sera modifiée  
    """  
    n = . . . . .  
    for i in range(. . . . ., . . . . .) :  
        valeur = tab[. . . . .]  
        j = . . . . .  
        while j >= . . . . . and tab[. . . . .] > . . . . . :  
            tab[. . . . .] = tab[. . . . .]  
            tab[. . . . .] = . . . . .  
            j -= . . . . .  
    return tab
```

2°) On appelle `tri_insertion([4, 5, 1, 9, 6, 2])`, donner l'état des différentes variables et de la condition d'arrêt de la boucle `while` (True ou False) lors de l'exécution de cette fonction :

Etat initial du tableau : `tab = [4, 5, 1, 9, 6, 2]`

BOUCLE FOR sur i			BOUCLE WHILE sur j				
			Début de boucle			Fin de boucle	
i	valeur	j	j	tab[j]	Condition de la boucle WHILE	tab	j
1	5	0	0	4	False	/	/
2	1	1	1	5	True	[4, 1, 5, 9, 6, 2]	0
/	/	/	0	4	True	[1, 4, 5, 9, 6, 2]	-1
/	/	/	-1	/	False	/	/
3							

Etat final du tableau : `tab = [..., ..., ..., ..., ..., ...]`

Exercice 3 – Recherche dichotomique (5 points)

1°) Effectuer à la main une recherche dichotomique du nombre **valeur = 15** dans la liste
tab = [1, 2, 3, 4, 5, 6, 9, 11, 12, 13, 14, 16, 17, 19, 21, 24, 26, 30] :

Indiquer l'indice et la valeur testée ainsi que le résultat du test :

indice	valeur	Résultat du test
8	12	Trop petit

2°) Effectuer à la main une recherche dichotomique du nombre **valeur = 9** dans la liste
tab = [1, 2, 3, 4, 5, 6, 9, 11, 12, 13, 14, 16, 17, 19, 21, 24, 26, 30] :

Indiquer l'indice et la valeur testée ainsi que le résultat du test :

indice	valeur	Résultat du test
8	12	Trop grand

3°) Compléter la fonction `recherche(tab)` ci-dessous :

```
def recherche(tab, valeur) :  
  
    """  
  
    Recherche dichotomique d'une valeur dans une liste  
  
    Renvoie True si la valeur est présente, False sinon  
  
    ATTENTION : la liste doit être triée dans l'ordre croissant  
  
    """  
  
    i_min = . . . . .  
    i_max = . . . . .  
  
    while . . . . . <= . . . . . :  
        i_milieu = . . . . .  
        valeur_milieu = tab[. . . . .]  
        if valeur == valeur_milieu : # on a trouvé !  
            return . . . . .  
        elif valeur < valeur_milieu : # on continue à gauche...  
            . . . . . = . . . . .  
        else : # on continue à droite...  
            . . . . . = . . . . .  
  
    return . . . . .
```

Exercice 4 – Algorithmes gloutons (7 points)

1°) Compléter la fonction `monnaie_v1(tab)` ci-dessous :

```
def monnaie_v1(somme, liste_valeurs) :  
    """  
    Algorithme glouton et boucle while  
    Renvoie la liste des pièces à rendre pour un nombre minimal  
de pièces rendus  
    Attention : liste_valeurs doit être triée dans l'ordre  
croissant  
    """  
    liste_resultat = []  
    indice = len(liste_valeurs) - 1  
    while . . . . . :  
        valeur = liste_valeurs[indice]  
        if somme >= valeur :  
            somme . . . . .  
            liste_resultat . . . . .  
        else :  
            indice . . . . .  
    return liste_resultat
```

2°) Compléter la fonction `monnaie_v2(tab)` ci-dessous :

```
def monnaie_v2(somme, liste_valeurs) :  
    """  
    Algorithme glouton et boucle for  
    Renvoie la liste des pièces à rendre pour un nombre minimal  
de pièces rendus  
    Attention : liste_valeurs doit être triée dans l'ordre  
croissant  
    """  
    liste_resultat = []  
    for valeur in liste_valeurs[::-1] :  
        n . . . . .  
        liste_resultat . . . . .  
        somme . . . . .  
    return liste_resultat
```

3°) Les lutins doivent aider le Père Noël à ranger les cadeaux dans des malles, puis mettre les malles sur le traineau. Ils souhaitent optimiser le rangement pour **avoir le moins de malles possible**... mais les malles ont une contrainte, la somme des tailles des cadeaux qui s'y trouvent **ne peut pas dépasser 50**.

Tous les cadeaux sont contenus dans une liste « liste_cadeaux » :
 liste_cadeaux = [C01, C02, C03, C04, C05, C06, C07, C08, C09, C10]

Les lutins proposent l'algorithme suivant :

- On initialise la liste des malles comme étant une liste vide
- On prend chaque cadeau de la liste des cadeaux un par un
- On parcourt les malles de la liste des malles une par une jusqu'à trouver une malle ayant assez de place pour y mettre le cadeau.
 - Si une telle malle est trouvée, on place le cadeau dedans
 - Si aucune malle n'est trouvée, on crée une nouvelle malle, on y place le cadeau, on ajoute cette malle à la liste des malles
- On renvoie la liste des malles

a) Les tailles des cadeaux étant données dans le tableau ci-contre, Donner le contenu de chaque malle en appliquant l'algorithme.

Cadeaux	Tailles
C01	4
C02	2
C03	2
C04	20
C05	2
C06	9
C07	5
C08	8
C09	10
C10	35

Malles	Liste cadeaux	Somme des tailles
M01		
M02		
M03		

b) La solution donnée par l'algorithme n'est pas optimale...
 Proposer un meilleur rangement de la liste des cadeaux précédente :

Malles	Liste cadeaux	Somme des tailles
M01		
M02		

c) On propose une amélioration de l'algorithme précédent en effectuant un tri descendant des cadeaux en fonction de leurs tailles avant d'appliquer l'algorithme.

Trouver une liste de 6 nouveaux cadeaux pour lesquels, l'algorithme modifié avec le tri initial des cadeaux par ordre descendant des tailles, trouve une répartition en 3 malles, alors qu'il existe une répartition optimale en 2 malles !

Cadeaux	Tailles
C01	
C02	
C03	
C04	
C05	
C06	