Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour consulter tous nos sujets d'annales et corrigés du Bac Général Maths + NSI, veuillez visiter :
bac-maths-nsi.fr/annales-sujets-et-corriges

Métropole - juin 2021 - sujet 1 (corrigé)

Exercice 1 (SQL)

1. Si les relations `Piece` et `Acteur` sont vides, il ne sera pas possible de renseigner les attributs `idPiece` et `idActteur` de la relation, car `Role.idPiece` fait référence à la clé primaire `Piece.idPiece` de la relation `Piece` et `Role.idActeur` fait référence à la clé primaire `Acteur.idActeur` de la relation `Acteur` qui sont vides toutes les deux.
2. Pour ajouter le rôle dans la table (ou relation) `Role` on peut saisir :

```
INSERT INTO Role(idPiece, idActeur, nomRole) VALUES (46721, 389761, 'Tartuffe')
```

3. La requête proposée remplace `'Américain'` et `'Britanique'` par `'Anglais'` pour toutes les entrées concernées de la table `Piece`.
4. Voici des requêtes possibles en `SQL` pour
   (a) Le nom et prénom des artistes nés après 1990 :

```
SELECT nom, prenom FROM Acteur WHERE anneNaiss > 1990
```

   (b) L'année de naissance du plus jeune artiste :

```
SELECT MAX(anneeNaiss) FROM Acteur
```

   (c) Le nom des rôles joués par l'acteur Vincent Macaigne :

```
SELECT nomRole FROM Role
INNER JOIN Acteur ON Role.idActeur = Acteur.idActeur
WHERE Acteur.nom = 'Macaigne' AND Acteur.prenom = 'Vincent'
```

   (d) Le titre des pièces écrites en Russe dans lesquelles l'actrice Jeanne Balibar a joué :

```
SELECT Piece.titre FROM Piece
INNER JOIN Piece ON Role.idPiece = Piece.idPiece
INNER JOIN Acteur ON Role.idActeur = Acteur.idActeur
WHERE Piece.langue = 'Russe' AND Acteur.nom = 'Balibar' AND Acteur.prenom = 'Jeanne'
```

Exercice 2 (Piles et POO)

1. (a) Les instructions suivantes conviennent :

```
pile1 = Pile()
pile1.empiler(7)
pile1.empiler(5)
pile1.empiler(2)
```

   (a) On a l'affichage suivant : `7, 5, 5, 2`

2. (a) On a les affichages suivants :

Cas 1	Cas 2	Cas 3	Cas 4
`3, 2`	`3, 2, 5, 7`	`3`	`pile vide`

   (b) La fonction `mystere` renvoie une pile qui contiendra tous les éléments de la pile `pile` passée en paramètre à condition qu'ils soient situés au-dessus de l'élément `element` passé en paramètre. L'élément `element` sera lui aussi présent dans la pile renvoyée par la fonction.

3. La fonction suivante convient :

```
def etendre(pile1, pile2):
    while not pile2.est_vide():
        x = pile2.depiler()
        pile1.empiler(x)
```

4. La fonction suivante convient :

```
def supprime_toutes_occurences(pile, element):
    p2 = Pile()
    while not pile.est_vide():
        x = pile.depiler()
        if x != element:
            p2.empiler(x)
    while not p2.est_vide():
        x = p2.depiler()
        pile.empiler(x)
```

Exercice 3 (Processus et routage)
Partie A : processus.

1. La première commande exécutée par le système d'exploitation lors du démarrage est la commande `init`.
2. Les identifiants des processus actifs sur cet ordinateur au moment de l'appel de la commande `ps` sont `5440` et `5450`, car ce sont les seuls processus ayant l'indicateur `R` dans la colonne `STAT`.
3. - La commande `ps` a la valeur `1912` pour `PPID`, qui correspond au `PID` de l'application `Bash`. La commande `ps` a donc été lancée depuis l'application `Bash`.
   - Deux autres commandes `Bash` (`PID` de `2014` et `2013`) et le programme `python programme1.py` (`PID` de `5437`) ont été lancés depuis l'application `Bash` de `PID` `1912`.
4. La commande `python programme1.py` a pour `PID` `5437` et la commande `python programme2.py` a pour `PID` `5440`. La commande `python programme1.py` a donc été lancée avant la commande `python programme2.py`.
5. Aucune prédiction ne peut être réalisée. Tout dépend de l'ordonnancement et des ressources à utiliser par les deux processus.

Partie B : routage.

1. On a le tableau suivant :

Machine	Prochain saut	Distance
A	D	3
B	C	3
C	E	2
D	E	2
E	F	1

2. On a les coûts suivants :

Liaison	A-D	A-B	B-C	C-D	C-E	D-E	E-F
Coût	10	1	1	10	1	10	1

On en déduit alors que l'on a le tableau suivant :

Machine	Prochain saut	Coût
A	B	4
B	C	3
C	E	2
D	E	11
E	F	1

3. RIP ne tient pas compte du débit alors que OSPF en tient compte. OSPF sera donc le plus performant.

Exercice 4 (Tableaux)

Partie A : représentation d'un labyrinthe.

1. Une instruction permettant de placer le départ au bon endroit dans `lab2` est : `lab2[1][0] = 2`
2. Il s'agit de vérifier que `i` et `j` sont des entiers positifs ou nuls inférieurs ou égaux respectivement à `n-1` et `m-1`.

```
def est_valide(i, j, n, m):
    return i >= 0 and j >= 0 and i < n and j < m
```

3. Il s'agit ici de parcourir la liste de listes (double boucle) jusqu'à trouver la valeur `2`.

```
def depart(lab) :
    n = len(lab)
    m = len(lab[0])
    for i in range(n):
        for i in range(m):
            if lab[i][j] == 2:
                return (i,j)
```

4. Il s'agit cette fois de compter le nombre de valeurs qui ne sont pas égales à 1 dans le tableau (0, 2, ou 3 indiquent des case vides).

```
def nb_cases_vides(lab):
    nb = 0
    for i in range(len(lab)):
        for j in range(len(lab[0])):
            if lab[i][j] != 1:
                nb += 1
    return nb
```

Partie B : recherche d'une solution dans un labyrinthe.

1. L'appel `voisines(1, 2, [[1, 1, 4], [0, 0, 0], [1, 1, 0]])` renvoie `[(1,1), (2,2)]`.
2. (a) En supposant que l'algorithme choisit d'abord de descendre dans la grille avant d'aller à droite, les instructions suivantes conviennent :

```
chemin.append((3, 3))
chemin.append((3, 4))
chemin.pop()
chemin.pop()
chemin.pop()
chemin.append((1, 4))
chemin.append((1, 5))
```

(b) Le code suivant convient :

```
def solution(lab):
    chemin = [depart(lab)]
    case = chemin[0]
    i = case[0]
    j = case[1]
    while lab[i][j] != 3: # tant que ce n'est pas l'arrivée
        lab[i][j] = 4 # on marque la case visitée
        voisins = voisines(i, j, lab)
        if voisins == [] : # cas de l'impasse
            chemin.pop()
            i,j = chemin[len(chemin)-1] # retour en arrière
        else:
            i, j = voisins[0]
            chemin.append((i,j))
    return chemin
```

Exercice 5 (Tableaux et récursivité)
Questions préliminaires.

1. A l'indice 1 du tableau on trouve 8 et à l'indice 3 on trouve 7.
   Comme $1 < 3$ alors que $8 > 7$, nous avons donc bien une inversion.
2. A l'indice 2 du tableau on trouve 3 et à l'indice 3 on trouve 7.
   Comme $2 < 3$ et $3 < 7$, nous n'avons donc pas d'inversion.

Partie A : méthode itérative.

1. (a)
   - ⋆ Cas 1 : `0`
   - ⋆ Cas 2 : `1`
   - ⋆ Cas 3 : `2`

   (b) Si on considère l'élément `b` situé à l'indice `i` dans le tableau `tab`. La fonction `fonction1` permet de déterminer le nombre d'éléments plus grands que `b` situés dans le tableau à un indice supérieur à `i`.
2. La fonction suivante convient :

```
def nombre_inversions(tab):
    nb_inv = 0
    n = len(tab)
    for i in range(n-1):
        nb_inv = nb_inv + fonction1(tab, i)
    return nb_inv
```

3. L'ordre de grandeur de la complexité en temps de l'algorithme est $O(n^2)$.

Partie B : méthode récursive.

1. Le tri fusion a une complexité en $O(n \log_2 n)$.
2. La fonction suivante convient :

```
def moitie_gauche(tab):
    n = len(tab)
    nvx_tab = []
    if n == 0:
        return []
    mil = n//2
    if n%2 == 0:
        lim = mil
    else :
        lim = mil+1
    for i in range(lim):
        nvx_tab.append(tab[i])
    return nvx_tab
```

3. La fonction suivante convient :

```
def nb_inversions_rec(tab):
    if len(tab) > 1:
        g = moitie_gauche(tab)
        d = moitie_droite(tab)
        return nb_inv_tab(tri(g),tri(d))+nb_inversions_rec(g)+nb_inversions_rec(d)
    else:
        return 0
```

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.