

2021 métropole candidat libre sujet 1

Exercice 1

1. [0.5 point] Car idPièce et idActeur sont des clés étrangères et doivent donc faire référence à des entrées des tables Piece et Acteur

2. [0.5 point] `INSERT INTO Role VALUES (46721, 389761, "Tartuffe");`

3. [0.5 point] Elle remplace dans la table Piece les valeurs « Américain » et « Britannique » du champ langue par « Anglais ».

4.a. [0.5 point] `SELECT nom, prenom FROM Acteur WHERE anneeNaiss >= 1990;`

4.b. [0.5 point] `SELECT MAX(anneeNaiss) FROM Acteur;`

4.c. [0.75 point]

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

4.d. [0.75 point]

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

Exercice 2

1.a. [0.5 point]

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

1.b. [0.5 point] 7, 5, 5, 2

2.a. [0.5 point]

1) 3, 2

2) 3, 2, 5, 7

3) 3

4) pile vide

2.b. [0.5 point] Elle extrait les éléments du sommet et les empile jusqu'à trouver l'élément passé en paramètre. Elle renvoie ensuite une pile des éléments extraits.

3. [0.75 point]

```
def etendre(pile1, pile2):
    nb_elements = pile2.nb_elements()
    for i in range(nb_elements):
        elem = pile2.depiler()
        pile1.empiler(elem)
```

4. [1.25 point]

```
def supprime_toutes_occurences(pile, element):
    pile2 = Pile()
    nb_elements = pile.nb_elements()
    for i in range(nb_elements):
        elem = pile.depiler()
        if elem != element:
            pile2.empiler(elem)
    nb_elements2 = pile2.nb_elements()
    for i in range(nb_elements2):
        pile.empiler(pile2.depiler())
```

Exercice 3

Partie A

1. [0.25 point] /sbin/init

2. [0.5 point] 5440 et 5450 car ils ont "R" dans leur stat.

3. [0.5 point] Bash. Les autres commandes sont Bash, Bash et "python programme1.py"

4. [0.25 point] python programme1.py en premier car elle a un PID plus petit.

5. [0.25 point] Non.

Partie B

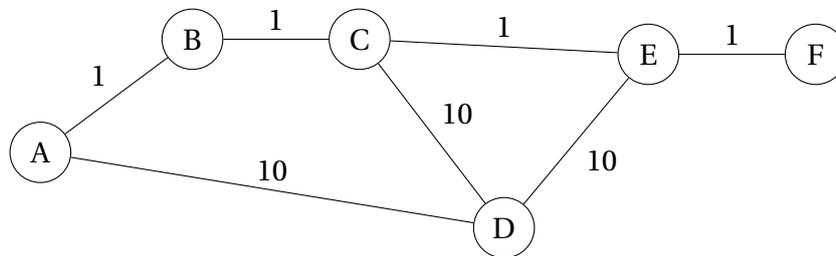
1. [0.5 point]

A	D	3
B	C	3
C	E	2
D	E	2
E	F	1

2. [1.25 point] On remplace les débits par les coûts sur le schéma :

$$100\text{Mbit/s} \rightarrow \frac{10^8}{10^8} = 1$$

$$10\text{Mbit/s} \rightarrow \frac{10^8}{10^7} = 10$$



On obtient ensuite la table de routage :

A	B	4
B	C	3
C	E	2
D	E	11
E	F	1

3. [0.5 point] Le débit global d'une liaison est le plus faible débit de la route. Avec RIP le débit est donc 10Mbit/s et avec OSPF 100Mbit/s. Ainsi OSPF est plus performant de ce point de vue.

Exercice 4

Partie A

1. [0.25 point] `lab2[1][0] = 2`

2. [0.5 point]

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

3. [0.75 point]

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

4. [0.75 point]

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

Partie B

1. [0.25 point] Nous avons le labyrinthe suivant :

```
[[1, 1, 4],
 [0, 0, 0],
 [1, 1, 0]]
```

Le retour est donc : [(1, 1), (2, 2)]

2.a. [0.5 point]

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

2.b. [1 point]

```
def solution(lab):
    chemin = [depart(lab)]
    case = chemin[0]
    i = case[0]
    j = case[1]
    while lab[i][j] != 3:
        lab[i][j] = 4
        if voisines(i, j, lab) != []:
            case = voisines(i, j, lab)[0]
            chemin.append(case)
        else:
            chemin.pop()
            case = chemin[-1]
            i = case[0]
            j = case[1]
    return chemin
```

Exercice 5

1. [0.25 point] Car $8 > 7$.

2. [0.25 point] Car $3 < 7$.

Partie A

1.a. [0.5 point]

1) 0

2) 1

3) 2

1.b. [0.5 point] Le nombre d'inversion avec i à gauche.

2. [0.5 point]

```
def nombre_inversion(tab):
    nb_elem = len(tab)
    cpt = 0
    for i in range(0, nb_elem - 1):
        cpt = cpt + fonction1(tab, i)
    return cpt
```

3. [0.25 point] $O(n^2)$

Partie B

1. [0.25 point] Tri fusion.

2. [0.5 point]

```
def moitie_gauche(tab):
    milieu = (len(tab) + 1) // 2
    return tab[:milieu]
```

3. [1 point]

```
def nb_inversions_rec(tab):
    if len(tab) <= 1: # Condition d'arrêt
        return 0
    else:
        tab_g = moitie_gauche(tab)
        tab_d = moitie_droite(tab)
        nb = nb_inversions_rec(tab_g) + nb_inversions_rec(tab_d)
        tab_g_trie = tri(tab_g)
        tab_d_trie = tri(tab_d)
        nb = nb + nb_inv_tab(tab_g_trie, tab_d_trie)
    return nb
```