**Sujet 2**

**Exercice 1 :**

1. Cette phrase décrit le comportement d’une file car le premier rentré est le premier sorti (first in, first out).
2. Les valeurs successives prises par la variable contrôleur lors de leur analyse sont les suivantes :

B : 1, 2 ,3, 2, 3, 2 🡪 Le parenthésage n’est pas correct.

C : 1, 2, 1, 0, -1, 0 🡪 Le parenthésage est correct (d’après la variable).

1. Voici les lignes de codes manquantes :

Ligne 13 : if controleur == -1 :

Ligne 16 : if controleur == 0 :

1. Voici les différentes étapes de la pile :  
   Etape 1 : []

Etape 2 : [‘<p>’]

Etape 3 : [‘<p>’, ‘<em>’]

Etape 4 : [‘<p>’]

Etape 5 : []

1. Si la pile finit par être vide, alors me balisage est correct.
2. Le nombre d’éléments que pourrait contenir au maximum la pile lors de son analyse est de 6.

**Exercice 2 :**

1. La requête renvoie le nom, le prénom et la naissance de l’individu qui a pour nom « Crog » donc cela renvoie : ‘Crog’ ‘Daniel’ ’07-04-1968’
2. Voici ma requête :

SELECT titre, id\_rea

FROM realisation

WHERE annee > 2020

1. Pour modifier la date de naissance de Daniel Crog, il faut utiliser la requête 1 car la requête 2 ajoute un élément (qui existe déjà) dans la table individu.
2. Oui, la table individu peut accepter deux individus ayant le même nom, le même prénom et la même date de naissance tant que l’id\_ind est différente à chaque fois car c’est la clé primaire.
3. Voici les requêtes :

INSERT INTO emploi

VALUES (5400, 'Acteur(James Bond)', 688, 105);

INSERT INTO emploi

VALUES (5401, 'Acteur(James Bond)', 688, 325);

1. On doit d’abord créer l’enregistrement du film dans la relation « realisation » car la relation « emploi » à besoin de l’id\_rea (l’identifiant du film) pour rajouter un nouvel emploi de Daniel Crog en tant que James Bond dans le film 'Docteur Yes'.
2. Voici la requête :

SELECT individu.nom, realisation.titre, realisation.annee

FROM emploi

JOIN individu ON emploi.id\_ind

JOIN realisation ON emploi.id\_rea

WHERE emploi.description = 'Acteur(James Bond)';

1. Voici la requête :

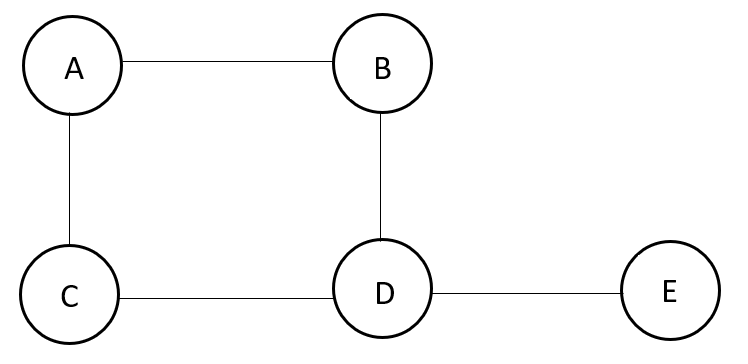
SELECT emploi.description

FROM emploi

JOIN individu ON emploi.id\_ind

WHERE individu.nom = ‘Johnson’ and individu.prenom = ‘Denis’;

**Exercice 3 :**

1. L’écriture décimale de l’adresse IPv4 correspond à : 192.168.128.131
2. Le nombre d’adresses différentes possibles du réseau A est de 256.
3. A est directement relié à B, C et D.
4. 
5. 10\*\*8 / 500 \* 10\*\*3 = 200

10 = 10\*\*8 / d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Débit** | 100 kbps | 500 kbps | 10 mbps | 100 mbps |
| **Métrique associée** | 1 000 | 200 | 10 | 1 |

1. Le chemin emprunté par un message d’un ordinateur du réseau F à destination d’un ordinateur du réseau I est F 🡪 H 🡪 J 🡪 K 🡪 I (coup de 13).

|  |  |
| --- | --- |
| **Destination** | **Métrique** |
| F | 0 |
| G | 8 |
| H | 5 |
| I | 13 |
| J | 6 |
| K | 8 |
| L | 11 |

1. Le routeur H ou la liaison de H tombe en panne

**Exercice 4 :**

**Partie A : Parcours d’un arbre**

1. 3 + 6 + 7 + 4 + 2 + 9 + 1 = 32

La somme de l’arbre précédent est 32.

1. Lettre correspondent à :

A = racine

B = nœud

C = feuille

D = SAG (sous arbre gauche)

E = SAD (sous arbre droite)

1. La proposition correcte est la proposition C.
2. def somme(une\_liste) :

if len(une\_liste) == 1 :

return une\_liste[0]

else :

nombre\_enlever = une\_liste.pop(-1)

une\_liste[-1] = une\_liste[-1] + nombre\_enlever

return somme(une\_liste)

1. Le type de parcours obtenu grâce à la fonction parcourir est le parcour en largeur.

**Partie B : Méthode ‘diviser pour régner’**

1. La seule proposition correcte est la proposition D.
2. somme(arbre) = valeur\_racine + somme(SAG) + somme(SAD)
3. def calcul\_somme(arbre) :

if arbe\_vide(arbre) :

return 0

else :

return valeur\_racine(arbre) + calcul\_somme(arbre\_gauche(arbre)) + calcul\_somme(arbre\_droit(arbre))

**Exercice 5 :**

1. La bonne instruction est l’instruction 3.
2. def redevenir\_actif(self) :

if self.est\_actif == False :

self.est\_actif = True

1. def nb\_tirs\_recus(self) :

return len(self.liste\_id\_tirs\_recus)

1. C’est le test 1 qui permet de vérifier qu’en fin de partie un participant égaré n’a pas rejoint par erreur la base adverse.
2. Le score de la base lorsqu’un joueur de cette équipe a été touché par le tir d’un coéquipier baisse de 20 points.
3. If participant.est\_determine()

self.incremente\_score(40)

E

D

C

B

A