Baccalauréat Général

Session 2022

Épreuve : **Numérique et sciences informatiques**

Durée de l'épreuve : 3 heures 30

Coefficient : 16

PROPOSITION DE CORRIGÉ

Exercice I :

Partie A

1) La phrase décrit le comportement d'une file puisque le premier élément entré est aussi le premier élément qui sera lu (first in first out).

2) Pour l'expression B, la variable contrôleur prend successivement les valeurs : 1, 2, 3, 2, 3, 2
Pour l'expression C, la variable contrôleur prend successivement les valeurs : 1, 2, 1, 0, -1, 0

3)

```
13                  if controleur < 0:
14                      #parenthèse fermante sans parenthèse ouvrante
15                      return(False)
16          if controleur ==0:
```

Partie B

4) a.

Pile

Pile

  « balise ouvrante, on empile »
Δ

Pile

  « balise ouvrante, on empile »
Δ

Pile

  « balise fermante, on dépile,
Δ   et correspondent,
on continue »

Pile

  « balise fermante, on dépile,
Δ   et correspondent,
on continue »

L'expression a été parcourue en entier le programme s'arrête, balisage correct.

b. Comme décrit dans le a., à la fin du parcours de l'expression si l'analyse de n'est pas arrêtée et si la pile est vide alors le balisage est correct.

5) Si une expression correctement balisée contient 12 balises, c'est qu'elle contient 6 balises ouvrantes et 6 balises fermantes. Le nombre d'éléments que pourraient contenir la

pile au maximum correspond au cas où l'expression est d'abord composée des 6 balises ouvrantes suivies des 6 balises fermantes. Cela signifie, si l'on suit l'algorithme donné dans l'énoncé que les 6 balises ouvrantes vont être d'abord empilées avant d'être dépilées. Dans ce cas, la pile contiendra au maximum 6 éléments.

Exercice II :

1) **a.** La requête renvoie les attributs nom, prenom et naissance des entités de la relation individu dont le nom est 'Crog' soit : 'Grog' 'Daniel' '07-07-1968'

b. SELECT titre,id_rea
FROM realisation
WHERE annee > 2020 ;

2) **a.** Il faut utiliser la requête 1 pour modifier la date de naissance de Daniel Grog. En effet, la requête 2 entrainerait une erreur dans la mesure où l'entité de la relation individu qui a pour clé primaire id_ind = 688 existe déjà donc il est impossible d'en créer une autre, la clé primaire est unique.
b. C'est possible dans la mesure où seule la clé primaire id_ind doit être unique. Donc la relation pourrait très bien contenir une deuxième entité avec les mêmes nom, prénom et date de naissance mais avec une clé primaire id_ind différente.

3) **a. INSERT INTO** emploi
VALUES (5400,'Acteur(James Bond)', 688, 105) ;
INSERT INTO emploi
VALUES (5401,'Acteur(James Bond', 688, 325) ;

b. Pour pouvoir ajouter une nouvelle entité dans la table emploi, il faut que les clés étrangères de cette nouvelle entité Id_ind et id_rea existent déjà comme clés primaires dans les tables individu et réalisation. Or la clé primaire du film 'Docteur

Yes' n'existe pas encore dans la table réalisation puisque le film n'a pas encore été créé dans cette table.
Il faut donc commencer par créer l'entité 'Docteur Yes' de ce film dans la table réalisation et ensuite seulement, il sera possible de créer l'entité demandée dans la table emploi.

4) a. SELECT individu.nom, realisation.titre, realisation.annee
FROM emploi
JOIN individu **ON** individu.id_ind=emploi.id_ind
JOIN realisation **ON** realisation.id_rea = emploi.id_rea
WHERE emploi.description = 'Acteur(James Bond)' ;

5) b. SELECT emploi.description
FROM emploi
JOIN individu **ON** individu.id_ind=emploi.id_ind
WHERE individu.nom = 'Johnson' **AND** individu.prenom = 'Denis';

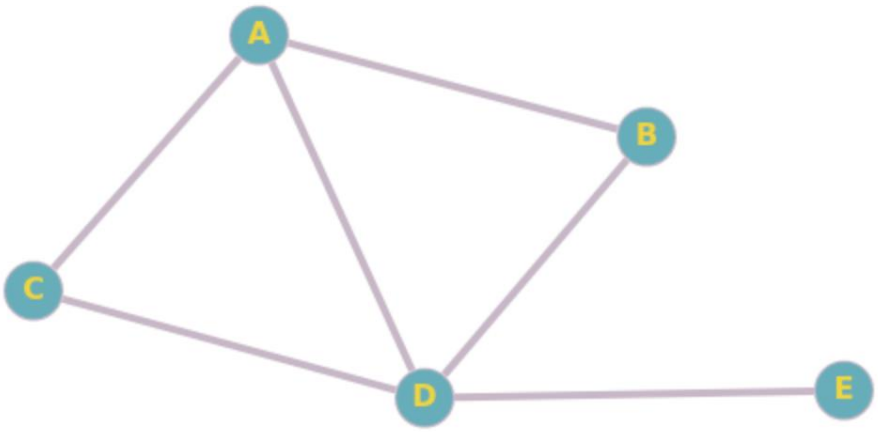
Exercice III :

1) a. 11000000 = 192 en base 10
10101000 = 168
10000000 = 128
10000011 = 131
L'adresse est donc **192.168.128.131**

b. Sur un octet il est possible d'avoir 2^8 =256 valeurs allant de 0 à 255 sachant que deux valeurs 0 et 255 sont réservées, il reste alors 254 valeurs à attribuer.

2) a. Les routeurs connectés à A sont à un saut, d'après la table B,C et D sont connectés à A

b.



3)

Débit	100 kbps	500 kbps	**10 Mbps**	100 Mbps
Métrique associée	1000	**200**	10	1

4) a. Le chemin le plus court en tenant compte de la métrique est : F→ H → J → K → I pour une métrique de 13

b.

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

c. La coupure de la liaison entre F et H de métrique 5 obligent les données à transiter par le router G.
En enlevant la connexion F-H tous les chemins les plus courts passent par G-F de métrique 10. L'autre possibilité est de passer par F-I de métrique 20 qui allonge trop le chemin par rapport aux autres liaisons.

Exercice IV :

Partie A

1) 3+6+2+7+4+9+1 = 32 utilisation d'un parcours en largeur.
Autre possibilité : En utilisant les sous-arbres de gauche et de droite 3 + (6 +(7+4))+(2+(9+1)) =
3+(6+11)+(2+10)=3+17+12=32

2) A : racine ; B : Nœud ; C : Feuille ; D : Sous-arbre gauche ; E : Sous arbre droit.

3) Le parcours en largeur correspond à la proposition C

4)

```
def somme(liste:list)->int:

somme=0

for n in liste:

    somme=somme+n

return somme
```

```
1 def somme(liste:list)->int:
2     somme=0
3     for n in liste:
4         somme=somme+n
5     return somme
6
7 somme([1,2,3,4])
```

Le parcours obtenu avec fonction parcourir est un **parcours en largeur** ;

Partie B :

6) Proposition D :

En informatique, le principe diviser pour régner signifie:diviser un problème en deux problèmes plus petits et indépendants.

7) Somme(A)= Valeur(racine)+ Somme(SAG)+ Somme(SAD)

8) def calcul_somme(arbre):

if est_vide(arbre):

return 0

else:

return valeur_racine(arbre)+calcul_somme(arbre_gauche(arbre))+calcul_somme(droit(arbre))

```
def calcul_somme(arbre):
    if est_vide(arbre):
        return 0
    else:
        return valeur_racine(arbre)+calcul_somme(arbre_gauche(arbre))+calcul_somme(droit(arbre))
```

Exercice V :

1) L'instruction 3 est la bonne façon de déclarer un objet.
joueur1 = Joueur("Sniper",319,"A")

2) a. def redevenir_actif(self):
if not self.est_actif :
self.est_actif = True

```
def redevenir_actif(self):
    if not self.est_actif :
        self.est_actif = True
```

b. def nb_de_tirs_recus(self):

return len(self.liste_id_tirs_recus)

```
def nb_de_tirs_recus(self):
    return len(self.liste_id_tirs_recus)
```

3) a. Le **test 1** vérifie l'appartenance d'un joueur à l'équipe
b. Le **test 2** vérifie si l'identifiant d'un joueur présent dans la liste des tirs est un coéquipier ou un joueur de l'équipe adverse. Si le joueur a été touché par un coéquipier le score de l'équipe baisse de 20 si le joueur a été touché par un adverse le score baisse de 10.

4)

```
if participant.est_détermine(): # si le participant réalise un grand nombre de tirs
    self.incremente_score(40) # le score de la base augmente de 40
```

if participant.est_détermine(): # si le participant réalise un grand nombre de tirs

self.incremente_score(40) # le score de la base augmente de 40