2022 Métropole sujet 1

Exercice 1

Partie A

1. [0.5 point] C'est le comportement d'une file car le premier arrivé est le premier sorti (FIFO).

2. [0.5 point]
- B : 0, 1, 2, 3, 2, 3, 2 ;
- C : 0, 1, 2, 1, 0, -1, 0.

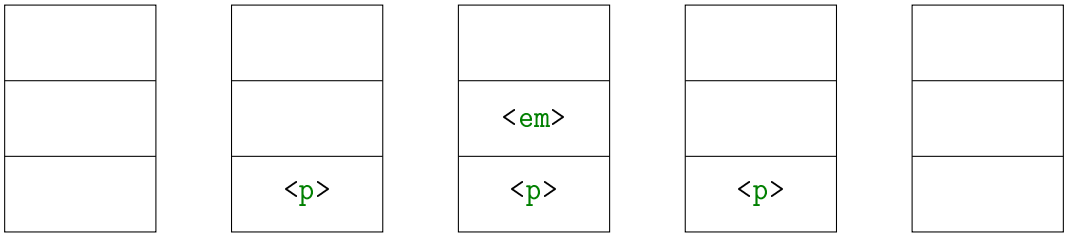
3. [1 point]

```
def parenthesage_correct(expression):
    """ fonction retournant True si l'expression arithmétique
    simplifiée (str) est correctement parenthésée, False
    sinon.
    Condition: expression ne contient que des parenthèses
    ouvrantes et fermantes """

    controleur = 0
    for parenthese in expression: #pour chaque parenthèse
        if parenthese == '(':
            controleur = controleur + 1
        else: # parenthese == ')'
            controleur = controleur - 1
            if controleur < 0 : # test 1 (à recopier et compléter)
                # parenthèse fermante sans parenthèse ouvrante
                return False
    if controleur == 0 : # test 2 (à recopier et compléter)
        return True # le parenthésage est correct
    else:
        return False # parenthèse(s) fermante(s) manquante(s)
```

Partie B

4.a. [1 point]



4.b. [0.5 point] Il faut que la pile soit vide.

5. [0.5 point] Pour obtenir le maximum, toutes les balises doivent être imbriquées avec 6 balises ouvrantes et 6 balises fermantes. Donc la pile pourrait contenir au maximum 6 éléments.

Exercice 2

1.a. [0.5 point]

```
+------+--------+------------+
| Crog | Daniel | 07-07-1968 |
+------+--------+------------+
```

1.b. [0.5 point]

```sql
SELECT id_rea, titre FROM realisation WHERE annee > 2020;
```

2.a. [0.5 point] Il faut utiliser la requête 1 pour faire une insertion. La requête 2 ne fonctionnera pas car la valeur 688 pour la clé primaire `id_int` est déjà présente dans la table.

2.b. [0.5 point] Oui, la relation `individu` peut accepter deux individus avec le même nom, le même prénom et la même date de naissance s'ils on un identifiant différent.

3.a. [0.5 point]

```sql
INSERT INTO emploi
VALUES (5400, 'Acteur(James Bond)', 688, 105);

INSERT INTO emploi
VALUES (5401, 'Acteur(James Bond)', 688, 325);
```

3.b. [0.5 point] Il faut d'abord créer l'enregistrement du film dans la relation `realisation` pour que la clé étrangère `id_rea` de la relation `emploi` ait une correspondance dans la relation `realisation`.

4.a. [0.5 point]

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

4.b. [0.5 point]

```sql
SELECT description
FROM emploi
JOIN individu ON emploi.id_ind = individu.id_ind
WHERE individu.prenom = 'Denis' AND individu.nom = 'Johnson';
```

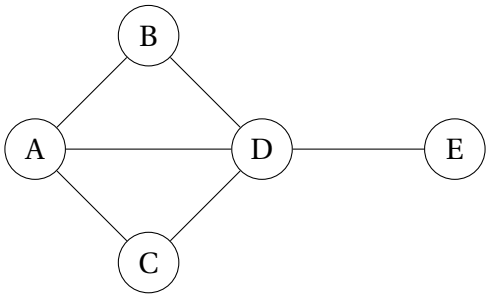
Exercice 3

1.a. [0.5 point] 192.168.128.131

1.b. [0.25 point] Il y a 256 adresses possibles dont une réservée pour l'adresse du réseau et une autre pour l'adresse de diffusion.

2.a. [0.25 point] B, C et D.

2.b. [1 point]



3. [0.5 point]

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

4.a. [0.5 point] Le chemin est F-H-J-K-I. C'est le chemin avec la somme des coût la plus faible : 13.

4.b. [0.75 point]

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

4.c. [0.5 point] Cela se produira si H tombe en panne car la métrique de la liaison F-I est grande. On peut alors écrire la nouvelle table de routage de F avec les chemins complets pour vérifier que tout passe par G :

Destination	Métrique	Chemin
F	0	F
G	10	F-G
H	-	-
I	19	F-G-J-K-I
J	12	F-G-J
K	14	F-G-J-K
L	17	F-G-J-L

Tous les chemins passent bien par G.

Exercice 4

Partie A

1. [0.5 point] $S = 3+6+2+7+4+9+1 = 32$

2. [0.5 point]
- A : racine ;
- B : nœud ;
- C : feuille ;
- D : SAG ;
- E : SAD.

3. [0.5 point] La C.

4. [0.5 point]

```
def somme(tab):
    s = 0
    for e in tab:
        s += e
    return s
```

5. [0.5 point] C'est un parcours en largeur.

Partie B

6. [0.25 point] La D.

7. [0.5 point]

```
somme(arbre) = valeur_racine + somme(SAG) + somme(SAD)
```

8. [0.75 point]

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

Exercice 5

1. [0.5 point] `joueur1 = Joueur("Sniper", 319, "A")`

2.a. [0.75 point]

```
def redevenir_actif(self):
    if self.est_actif == False:
         self.est_actif = True:
```

2.b. [0.75 point]

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

3.a. [0.5 point] Test 1.

3.b. [0.5 point] Le score de l'équipe diminue de 20.

4. [1 point]

```
    if participant.est_determine():
        self.incremente_score(40)
```