Centres étrangers - mai 2022 - sujet 2 (corrigé)

Exercice 1 (Programmation générale et récursivité)

1. (a) La commande `f(5)` affiche :

```
5
4
3
2
1
Partez!
```

   (b) La fonction est récursive car elle s'appelle elle-même.

2. (a) On a le code suivant :

```
def ajouter(s, liste):
    res = []
    for m in liste:
        res ....................
    return res
```

   (b) La commande `ajouter("b", ["a","b","c"])` renvoie `['ba', 'bb', 'bc']`.

   (c) La commande `ajouter("a", [""])` renvoie `['a']`.

3. (a) La commande `produit("ab", 0)` renvoie `['']`, qui n'est pas une liste vide, mais une liste contenant une chaine de caractère vide.

   (b) La commande `produit("ab", 1)` renvoie `['a', 'b']`.

   (c) La commande `produit("ab", 2)` renvoie `['aa', 'ab', 'ba', 'bb']`.

Exercice 2 (Dictionnaires)

1. (a) La valeur associée à la clé `"D"` est `"C"`. Il faut saisir l'instruction `alpha["D"]`.

   (b) La chaîne de caractères `"BAGAGE"` est chiffrée par `"DBEBEF"`.

2. La fonction suivante convient :

```
def chiffrer(mot, alpha):
    mc = ""
    for l in mot:
        mc = mc + alpha[l]
    return mc
```

3. (a) Il faut échanger les clés et les valeurs. Un dictionnaire permettant de déchiffrer est donc :

```
alpha_d ={"B":"A", "D":"B", "A":"C", "C":"D", "F":"E", "G":"F", "E":"G"}
```

   (b) La fonction suivante convient :

```
def dico_dechiffrement(dico):
    nouveau = {}
    for lettre in dico :
        code = dico[.....................]
        nouveau[.....................] = .....................
    return nouveau
```

   (c) La fonction suivante convient :

```
def dechiffre(mot, dico):
    dico_d = dico_dechiffrement(dico)
    md = chiffrer(mot, dico_d)
    return md
```

4. La fonction suivante convient :

```
>>> tab = ["A", "B", "C", "D"]
>>> shuffle(tab)
>>> tab
["B", "A", "D", "C"]
```

Exercice 3 (Bases de données et SQL)

1. (a) La clé primaire devant être unique, le seul attribut que peut être unique pour chaque entrée, est l'attribut `Code_evaluation`. Par conséquent, le seul attribut qui peut jouer le rôle de clé primaire est `Code_evaluation`.

   (b) La requête suivante convient :

```
INSERT INTO Evaluations
VALUES ('EXKVLX886', 'Term7', 'Peltier', '13/10/2021', 1453)
```

2. (a) La requête sélectionne tous les auteurs (y compris ceux qui sont identiques) dans la table `Evaluations`. Elle renvoie donc onze lignes.

   (b) Les lignes issues de la requête sont :

```
SELECT Nom_evaluation, Date FROM Evaluations WHERE auteur= "Peltier"
```

   (c) La requête suivante convient :

```
SELECT Nom_evaluation
FROM Evaluations
WHERE Code_competences = 452
```

3. (a) Il faut que le couple `(Code_evaluation, Num_eleve)` soit unique. Un élève donné ne peut donc pas faire plusieurs fois la même évaluation.

   (b) La requête suivante convient :

```
SELECT Num_eleve
FROM resultats
JOIN Evaluations ON resultats.Code_evaluation = Evaluations.Code_evaluation
WHERE Code_competences = 532
```

4. (a) On a la structure suivante :

attribut	type
Num_eleve	INT
Nom	CHAR
Prenom	CHAR
Classe	CHAR

   (b) L'attribut `Num_eleve` peut jouer le rôle de clé primaire.

Exercice 4 (POO)

1. (a) Le code suivant convient :

```
class Carte:
    def __init__(self, val, coul):
        .......... .valeur = ..........
        ..................... = coul
```

   (b) Il faut saisir l'instruction `c7 = Carte(7, "coeur")`.

2. Le code suivant convient :

```
def initialiser() :
    jeu = []
    for c in ["coeur", "carreau", "trefle", "pique"] : # couleur carte
        for v in range(...) : # valeur carte
            carte_cree = ...
            jeu.append(carte_cree)
    return jeu
```

3. La structure des données la plus adaptée est la file, puisque l'on a affaire à une structure de type FIFO (First IN First OUT). Le classement des cartes doit suivre la « règle FIFO », car la carte remportée (la dernière arrivée) doit être placée en dessous du tas.

4. La fonction suivante convient :

```
def comparer(carte1, carte2):
    if carte1.valeur > carte2.valeur :
        return 1
    elif carte1.valeur < carte2.valeur :
        return -1
    else :
        return 0
```

Exercice 5 (Architecture matérielle, OS et réseaux)

1. (a) Puisque $31 = 2^4 + 2^3 + 2^2 + 2^1 + 2^0$, la conversion de `O4` en binaire est `00011111`.

   (b) La notation `/24` signifie que les 24 bits de poids forts sont égaux à 1 et les 8 bits restants sont égaux à 0. On en déduit que :
   - la notation binaire est `11111111 . 11111111 . 11111111 . 00000000`;
   - la notation décimale est `255.255.255.0`.

   (c) Il est possible d'adresser $256 - 2 = 254$ machines. En effet, deux adresses ne sont pas disponibles : adresse du réseau et adresse de broadcast.

2. (a) L'adresse IP `70.37.150.6` (renseignée dans la table de routage de R4) correspond au routeur R6, le routeur R4 envoie donc le paquet de données vers le routeur R6.

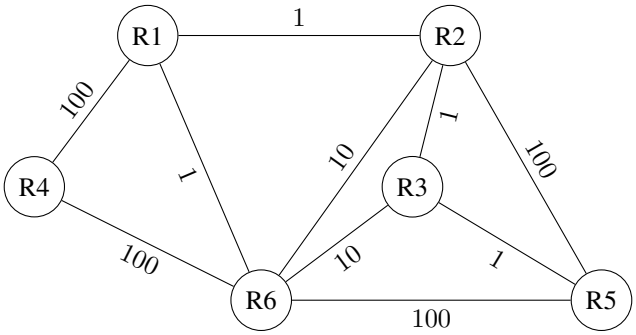
   (b) Le paquet traverse successivement les routeurs R4, R6 et R5.

3. (a) Le paquet peut suivre le chemin R4 – R1 – R2 – R5 ou R4 – R1 – R6 – R5.

   (b) Il faut modifier la ligne du routeur R4 de la façon suivante :

Routeur	Réseau destinataire	Passerelle	Interface
R4	`192.168.10.0`	`144.50.65.1`	`144.50.65.4`

4. (a) On a le schéma suivant :



   Le chemin parcouru est : R4 – R1 – R2 – R3 – R5 (coût de $100 + 1 + 1 + 1 = 103$).

   (b) Il faut modifier les lignes des routeurs R2, R4 et R6 comme suit :

Routeur	Réseau destinataire	Passerelle	Interface
R2	`192.168.10.0`	`85.40.65.3`	`85.40.65.2`
R4	`192.168.10.0`	`144.50.65.1`	`144.50.65.4`
R6	`192.168.10.0`	`32.18.145.3`	`32.18.145.6`