Éléments de correction
sujet 10

Exercice 1

1.

| Table de routage R1 | | |
|---|---|---|
| destination | prochain saut | distance |
| R2 | R2 | 0 |
| R3 | R2 | 1 |
| R4 | R4 | 0 |
| R5 | R5 | 0 |
| R6 | R5 | 1 |

2. R1 -> R5 -> R6
3.

| Table de routage R1 | | |
|---|---|---|
| destination | prochain saut | coût |
| R2 | R2 | 10 |
| R3 | R2 | 11 |
| R4 | R2 | 12 |
| R5 | R2 | 13 |
| R6 | R2 | 14 |

4. R1 -> R2 -> R3 -> R4 -> R5 -> R6
5. R1-> R5 -> R6
6.

| Calcul d'une adresse de réseau | | | | |
|---|---|---|---|---|
| machine | 11000000 | 10101000 | 00000001 | 00110010 |
| masque | 11111111 | 11000000 | 00000000 | 00000000 |
| réseau | 11000000 | 10000000 | 00000000 | 00000000 |
| réseau (déc) | 192 | 128 | 0 | 0 |

7.

| Calcul d'une adresse de réseau | | | | |
|---|---|---|---|---|
| réseau | 11000000 | 10000000 | 00000000 | 00000000 |
| masque | 11111111 | 11000000 | 00000000 | 00000000 |
| comp masque | 00000000 | 00111111 | 11111111 | 111111111 |
| broadcast | 11000000 | 10111111 | 11111111 | 11111111 |
| broadcast (déc) | 192 | 191 | 255 | 255 |

8.

- 172.16.0.0
- 172.16.255.255
- $2^{16}$ - 2 = 65534 adresses

9.

```
def masquer(self, masque: str)->str:
    tmp = []
    ip = self.octets()
    crible = IPv4(masque).octets()
    for i in range(4):
        tmp.append(ip[i] & crible[i])
    return ".".join([str(element) for element in tmp])
```

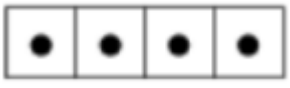
10.

```
def adresse_suivante(self, adresse_max:str)->str:
    assert self.adresse < adresse_max
    liste_courante = self.octets()
    liste_suivante = list()
    retenue = 1
    for index in range(4):
        somme = liste_courante[3 - index] + retenue
        valeur, retenue = somme%256, somme//256
        liste_suivante = = [str(valeur)] + liste.suivante
    return '.'.join(liste_suivante)
```

Exercice 2

1.

On peut placer un pion dans la première case :



On peut enlever un pion dans la troisième case :

2.

```
def initialiser(n):
    return [False]*n
```

3.

```
def victoire(tab):
    for etat_case in tab:
        if etat_case == False:
            return False
    return True
```

4.

```
def indice_premiere_case_occupee(tab):
    for i in range(len(tab)):
        if tab[i]:
            return i
    return None
```

5.

```
def coup_valide(tab, case):
    if case==0:
        return True
    elif case==indice_premiere_case_occupee(tab)+1 and case<len(tab) and case>=0:
        return True
    else:
        return False
```

6.

```
def changer_case(tab, case):
    if coup_valide(tab, case):
        tab[case] = not tab[case]
    return tab
```

7.

```
def vider(n):
    if n == 1:
        print('Vider case 1')
    elif n > 1:
        vider(n-2)
        print('Vider case ' + str(n))
        remplir(n-2)
        vider(n-1)
```

8.

Vider case 1
Vider case 3
Remplir case 1
Vider case 2
Vider case 1

9.
```
def remplir(n):
    if n == 1:
        print('Remplir case 1')
    elif n > 1:
        remplir(n-1)
        vider(n-2)
        print('Remplir case '+str(n))
        remplir(n-2)
```
10. Non, car il va y avoir trop d'appels récursifs (RecursionError, dépassement de la pile d'appels)

Exercice 3

1.
```
gen_mdp(8, True, True, False)
```
2.
```
def gen_mdp(longueur, cont_min, cont_maj, cont_spe):
    assert (cont_min or cont_maj or cont_spe)
    minuscules = [chr(i) for i in range(97, 123)]
    majuscules = [chr(i) for i in range(65, 91)]
    caracteres_speciaux = [chr(i) for i in range(33,48)] + [chr(i) for i in range(58, 65)]
```
3.
```
jeu_caracteres = []
if cont_min:
        jeu_caracteres += minuscules
        if cont_maj:
            jeu_caracteres += majuscules
        if cont_spe:
            jeu_caracteres += caracteres_speciaux
```
4.
```
mot_de_passe = ''
n = len(jeu_caracteres)
for i in range(longueur):
     mot_de_passe = mot_de_passe + jeu_caracteres[randint(0, n-1)]
return mot_de_passe
```
5. La fonction tire au sort les caractères, rien ne garantie donc que des caractères de chaque catégorie seront utilisés.
6. mot_de_passe est une clé primaire, le mot de passe doit donc être unique pour chaque entrée, Alice ne peut donc pas avoir 2 fois le même mot de passe.
7.
```
SELECT url
FROM site
```
8.
```
UPDATE compte
SET mot_de_passe = 'yhTS?d@UTJe'
```

```
WHERE mot_de_passe = '@rDfohpj!&'
```

9.
```
SELECT id_site
FROM compte
WHERE renouvellememement < '2024-03-20'
```

10. Permet d'ordonner les résultats par date de renouvellement, car l'ordre lexicographique correspond à l'ordre chronologique

11.
```
SELECT utilisateur, mot_de_passe
FROM compte
JOIN site ON id_site = id
WHERE nom_site = 'Votremailp'
ORDER BY renouvellement
```

12. On évite la redondance des informations

13.
```
chiffrement('gestionnaire.db','../Perso/secret.db','../Perso/cle')
```

14. FA

15.

| a | b | a XOR b | (a XOR b) XOR b |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

Donc a (a XOR b) XOR b = a quelque soit b

16. chiffrement symétrique car elle utilise la même clé pour le chiffrement et le déchiffrement.

17. La lecture est autorisée pour tout le monde, cela peut entrainer des problèmes de sécurité. Pour résoudre ce problème, il suffit donc de supprimer le droit en lecture pour "tout le monde".

18. - le mot de passe est une clé primaire, il est donc différent pour chaque site
    - on peut définir la longueur du mot de passe en appelant `gen_mdp`.
    - il faut établir des droits corrects pour les fichiers