Éléments de correction
sujet 04

Exercice 1

1. `x = 0 ; y = 20`
2. Un fichier .py est avant tout un fichier texte et il est possible d'exécuter un fichier Python, donc tout programme Python peut être vu comme une chaine de caractères.
3. programme3 : termine ; programme4 :ne termine pas
   programme5 : termine ; programme6 : ne termine pas
4. Non, car si le programme ne termine pas, on ne renvoie pas False (reste "bloqué" au niveau du exec)
5. L'algorithme de Boyer-Moore permet de rechercher un motif dans un texte. La connaissance préalable du motif recherché permet d'éviter de comparer chaque lettre du texte avec le motif (algo naif) et donc de gagner en efficacité
6.
```
def arret_essai2(programme):
    return not recherche('while',programme)
```
7.
   - avec un appel récursif sans cas de base, on obtient une "boucle" infinie sans la présence d'un while
   - un programme qui contient une boucle while peut très bien s'arrêter (exemple : programme3)
8.
```
def terminaison_inverse(programme):
    if arret(programme):
        boucle_infinie
```
9. programme_paradoxal termine si et seulement si l'appel terminaison_inverse(programme_paradoxal) termine et cela si et seulement si programme_paradoxal ne termine pas. Finalement programme_paradoxal ne peut ni terminer, ni ne pas terminer. Ceci est absurde.
10. la fonction `arret` ne peut pas exister
11. Non, le problème de l'arrêt fait partie des problèmes indicidables : il n'existe pas d'algo capable de le résoudre (quel que soit le langage)

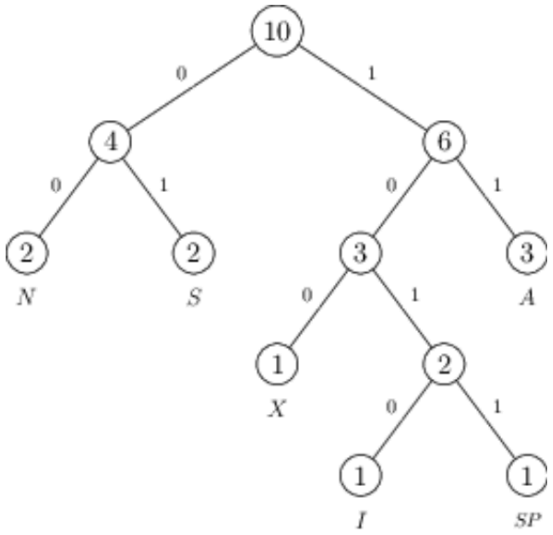
Exercice 2

1. 10 lettres correspond à 10 octets soit 80 bits
2. En hexa : 53 49 58 20 41 4E 41 4E 41 53
3.

symbole	S	I	X	A	N
Nbre occu	2	1	1	3	2

4. Au nombre total de caractères (taille du texte)

5.
```
def occurrence(texte):
    dico = {}
    for lettre in texte :
        if lettre in texte :
            dico[lettre] = dico[lettre]+1
        else:
            dico[lettre] = 1
    return dico
```
6.



7. Cela correspond à la taille du texte
8. Parcours en profondeur
9.

symbole	S	I	X	A	N	SP
Codage	01	1010	100	11	00	1011

10. Pour coder I on utilise 4 bits alors que pour coder S on utilise 2 bits
11. 01 1010 100 1011 11 00 11 00 11 01
12. (80 - 25) / 80 = 69 % ce qui est bien compris entre 20% et 90%

Exercice 3

1.
```
SELECT id-kimono
FROM location
WHERE fin = ''
```
2.
```
SELECT COUNT(id-kimono)
FROM kimono
WHERE taille-kimono = 130
```

3.
```
SELECT nom, prenom
FROM adherent
JOIN location ON adherent.numero-licence = location.numero-licence
WHERE id-kimono = 42 AND fin = ''
```
4.
```
UPDATE adherent
SET taille-adherent = taille-adherent + 10
WHERE taille-adherent < 160
```
5.
```
DELETE FROM location
WHERE id-kimono = 25

DELETE FROM kimono
WHERE id-kimono = 25
```
6.
M12102021NIRRE01
7.
Date de naissance : 23/09/1974 ; nom possible : MARTIN
8.
STEPHANIE
9.
```
tab_adherents[0]['numero-licence']
```
10.
```
def nombre_adherents(table, annee):
    compteur = 0
    for adherent in table:
        if adherent['annee'] == annee:
            compteur += 1
    return compteur
```
11.
```
def adherent_plus_age(table):
   mini_annee = 2024
   for adh in table:
      if adh['annee'] < mini_annee:
           lst = [adh]
           mini_annee = adh['annee']
      elif adh['annee'] == maxi_annee:
           lst.append(adh)
   return lst
```

12.

```
def verification_licence(adherent):
    licence = adherent['numero_licence']
    sexe = extraire(licence, 0, 1)
    if sexe != adherent['sexe']:
        return False
    date_nai_lic = extraire(licence, 1, 9)
    date_nai = adherent['jour']+adherent['mois']+adherent['annee']
    if date_nai_lic != date_nai :
        return False
    nom_lic = extraire(licence, 9, 14)
    nom = adherent['nom']
    nom_tr = ''
    for i in range(0,5):
        nom_tr = nom_tr + nom[i]
    if nom_lic != nom_tr :
        return False
    return True
```