Exercice 1 — 6 points

Questions	*Contenu et notions*	*Capacités exigibles / Niveau*	*Éléments de réponses et commentaires*
1	Analyse d'un programme Python	N1	`x = 0` et `y = 20`
2	Question de cours	N2	Un programme Python est un texte, par exemple dans un fichier `.py`. On peut donc considérer la chaîne de caractères correspondant à ce texte.
3	Analyse d'un programme Python	N2	Les programmes 3 et 5 terminent alors que les programmes 4 et 6 ne terminent pas.
4	Analyse d'un programme Python	N2	Cette fonction exécute le programme dont le code source est passé en paramètre. Une fois son exécution terminée elle renvoie `True`. Si le programme s'arrête, cette fonction renvoie correctement `True`. Mais si le programme ne s'arrête pas, cette fonction ne s'arrête pas non plus. Ceci ne répond donc qu'à moitié à la question.
5	Question de cours	N2	L'algorithme de Boyer-Moore permet de rechercher un motif dans un texte. Comme l'algorithme naïf il teste successivement les positions possibles du motif dans le texte, mais permet parfois d'effectuer des décalages de cette position plus grands que 1. Pour cela on compare les caractères du motifs et du texte à la position donnée, de droite à gauche. En cas d'échec d'une comparaison de caractères, on utilise une table de décalages pré-calculée pour connaître le meilleur décalage à effectuer.
6	Programmation en Python	N2	`def arret_essai2(programme):` `  return not recherche("while", programme)`

Exercice 1	6 points		

7		N3	On considère le programme `programme_rec` ci-dessous qui ne s'arrête pas alors que `arret_essai2(programme_rec)` renvoie `True`.
			`programme_rec = """`
			`def f():`
			`    return f()`
			
			`f()`
			`"""`
			Inversement, le programme `programme1` s'arrête mais `arret_essai2(programme1)` renvoie `False` (remarquons que l'on peut aussi considérer le programme `"z = 'while'"`, ici on ne garantit pas que l'on utilise effectivement `while` comme mot-clé d'une boucle).

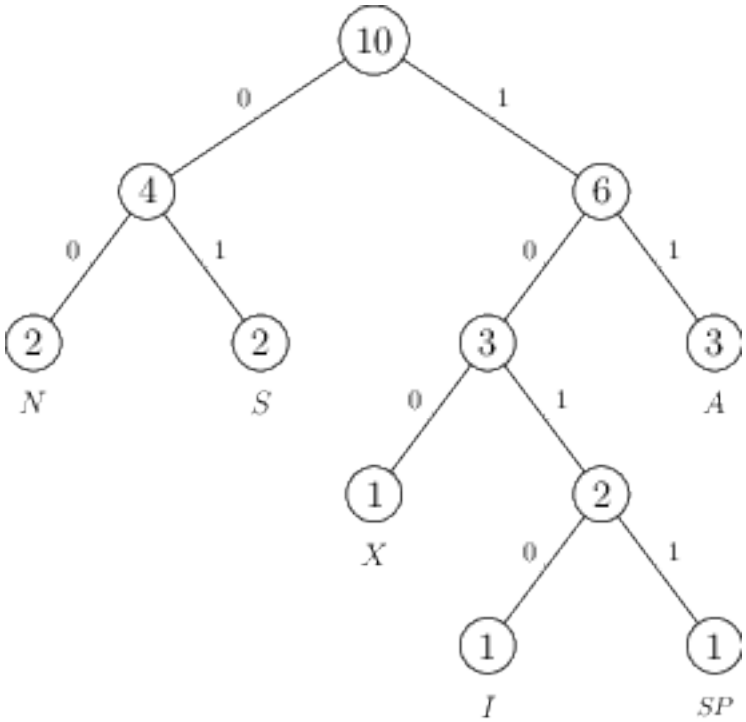
8	Programmation en Python	N3	`def terminaison_inverse(programme):`
			`    if arret(programme):`
			`        boucle_infinie()`

9	Analyse d'un problème	N3	`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	Analyse d'un problème	N2	La fonction `arret` ne peut pas exister.
11	Question de cours	N2	L'imposibilité d'écrire cette fonction ne dépend pas de Python.

Exercice 2	6 points		
Questions	*Contenu et notions*	*Capacités exigibles*	*Éléments de réponses et commentaires*
1	Connaitre la codage ASCII des caractères. Le nombre de bits d'un octet	Représentation d'un texte en machine.	txt comporte 10 caractères (incluant l'espace), chaque caractère est codé sur 1 octet soit taille = 10 octets = 10 x 8 = 80 bits
2	Passer de la représentation d'une base dans une autre. Convertir un fichier texte dans différents formats d'encodage.	N2	Par lecture de la table fournie, on a les informations du tableau 1. Ainsi, la variable `txt` est codée par : 0b 01010011 010001001 01011000 00100000 01000001 01001110 01000001 01001110 01000001 01010011 La notation hexadécimale sera également acceptée.
3	Compter le nombre d'occurences d'un symbole dans un texte.	N1	Voir le tableau 2.
4	Comprendre une définition. Faire une somme.	N1	La somme des nombres d'occurrence est égale à la taille du texte.

5	Coder en python. Itérer sur les éléments d'un dictionnaire.	N2	`def occurrence(texte):`
			`  dico = {}`
			`  for lettre in texte:`
			`  if lettre in dico.keys():`
			`  dico[lettre]+=1`
			`  else:`
			`  dico[lettre]=1`
			`  return dico`

Exercice 2 — 6 points



Correction Arbre Huffman pour “SIX ANANAS”

6	Analyser et modéliser un problème en termes de flux et de traitement d'informations. Réutiliser des codes sources existants.	N3	
7	Comprendre un modèle.	N1	Le poids de la racine est égale à la taille du texte.

Exercice 2	6 points		
8	Parcourir un arbre de différentes façons (ordres infixe, préfixe ou suffixe ; ordre en largeur d'abord).	N1	Il convient d'utiliser un parcours en profondeur.
9	Parcourir un arbre de différentes façons (ordres infixe, préfixe ou suffixe ; ordre en largeur d'abord).	N2	Voir le tableau 3.
10	décomposer un problème en sous-problèmes, reconnaître des situations déjà analysées et réutiliser des solutions ;	N2	Contrairement au codage ASCII où chaque symbole est codé sur une taille fixe de 8 bits, le codage Huffman code chaque symbole sur un bombre de bits différents.
11	Lecture d'un tableau	N1	SIX ANANAS se code donc 01 1010 100 1010 11 00 11 00 11 01
12	Calcul mathématique	N2	À l'aide du codage Huffman, txt se code sur 25 bits au lieu de 80. Ainsi t= (80-25)/80 x 100 = 68,75%. Cela correspond à la fourchette donnée dans le texte puisque $20 < 68{,}75 < 90$.

Tableau 1.

caractère	I	X	A	N	S	espace
code hexadécimal	0x49	0x58	0x41	0x4E	0x53	0x20
code binaire	0b010001001	0b01011000	0b01000001	0b01001110	0b01010011	0b00100000

Tableau 2.

symbole	espace	I	X	A	N	S
nombre d'occurence	1	1	1	3	2	2

Tableau 3.

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

Exercice 3 — 8 points

Questions	*Contenu et notions*	*Capacités exigibles / Niveau*	*Éléments de réponses et commentaires*
1	Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données	N1	`SELECT id_kimono FROM location WHERE fin = ```
2	Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données	N1	`SELECT COUNT (id_kimono) FROM kimono WHERE taille = 130`

3	Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données	N2	`SELECT adherent.nom, adherent.prenom`
			`FROM location JOIN adherent ON location.numero_licence = adherent.numero_licer`
			`WHERE location.id_kimono = 42 AND location.fin = ''`

4	Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données	N2	`UPDATE adherent`
			`SET taille_adherent = taille_adherent + 10`
			`WHERE taille_adherent < 160`

5	Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données	N3	`DELETE FROM location`
			`WHERE id_kimono = 25`
			`DELETE FROM kimono`
			`WHERE id_kimono = 25`

6	Comprendre une structuration de données	N1	M12102021NIRRE01
7	Comprendre une structuration de données	N1	date de naissance : 23 septembre 1974 et nom possible : martin.

Exercice 3	8 points		
8	Dictionnaires, index et clé	N1	'STEPHANIE'
9	Dictionnaires, index et clé	N1	tab_adherents[0]['numero_licence']
10	Parcours d'une structure linéaire - occurences	N1	

```
def nombre_annee(table, annee):
    """
    entree : (table) tableau de dictionnaire
             (annee) nombre entier
    sortie : nombre entier
    """
    compteur = 0
    for dictionnaire in table:
        if dictionnaire['annee'] == annee:
            compteur = compteur + 1
    return compteur
```

11	Dictionnaires, index et clé - recherche de maximum	N3	

```
def adherent_plus_age(table):
    """
    entree : (table) tableau de dictionnaires
    sortie : (liste_plus_age) liste de dictionnaires
    """
    liste_plus_age = []
    annee_min = table[0][annee]
    for dictionnaire in table:
        if dictionnaire[annee] < annee_min:
            liste_plus_age = [dictionnaire]
            annee_min = dictionnaire[annee]
        if dictionnaire[annee] == annee_min:
            liste_plus_age.append(dictionnaire)
    return liste_plus_age
```

Exercice 3	8 points		

12	Mise au point des programme	N3	On donne ici une solution sans les tranches.
			`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`