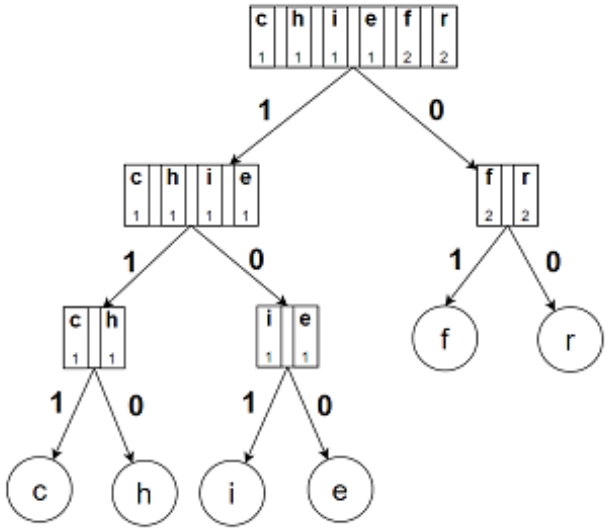
Exercice 1 (6 points)

Correction

Question	*Niveau*	*Contenu*	*Solution*
1	1	Arbres : structures hiérarchiques	`010`
2	1	Arbres : structures hiérarchiques	`espion`
3	1	Parcourir un arbre de différentes façons	Un parcours en largeur.
4	1	Présenter un problème ou sa solution	$1+1+1+1+1+1+1+2+2=11=3+4+4$
5	1	Évaluer quelques mesures	La hauteur de l'arbre est de 5 ce qui représente le nombre maximum de bits qui seront utilisés pour coder un symbole.

Question	*Niveau*	*Contenu*	*Solution*
		des arbres binaires	
6	2	1ère : Convertir un fichier texte dans différents formats d'encodage.	En ASCII nous aurons besoin de 22 octets car il y a 22 symboles dans l'expression. Avec la méthode de Shannon-Fano nous aurons besoin de $2 \times 5 + 9 \times 4 + 7 \times 3 + 4 \times 2 = 75$ bits donc 10 octets suffisent. D'autres raisonnements seront acceptés comme le simple fait d'expliquer qu'en moyenne les lettres sont codées par 4 bits soit un demi octet.
7	2	Arbres : structures hiérarchiques.	 Plusieurs arbres peuvent correspondre au problème, il suffit que f et r soient

Question	*Niveau*	*Contenu*	*Solution*
			de profondeur 2 et les 4 autres lettres de profondeur 3.
8	1	Dictionnaires, index et clé.	`8  dico[symbole] = dico[symbole] + 1` `10 dico[symbole] = 1`

9	2	1ère : Parcours séquentiel d'un tableau	`1 def somme_occ(tab):`
			`2     s = 0`
			`3     for elt in tab:`
			`4         s = s + elt[1]`
			`5     return s`

10	2	Écrire un programme récursif.	`9  return "1" + shannon(symbole, t1)` `11 return "0" + shannon(symbole, t2)`
11	2	Analyser le fonctionnement d'un programme récursif.	Par construction, à chaque appel de la fonction récursive le tableau passé en paramètre est de taille strictement inférieur au précédent et donc il finira par être de taille 1.

12	3	Utiliser la mémoïsation pour écrire un	`1 def encode_shannon(texte):`
			`2     dico = creer_dico_occ(texte)`
			`3     tab = creer_tab_trie(dico)`
			`4     dico_shannon = {}`
			`5     for symbole in dico:`
			`6         dico_shannon[symbole] = shannon(symbole, tab)`

Question	*Niveau*	*Contenu*	*Solution*
		algorithme .	7 code = '' 8 for caractere in texte: 9 code = code + dico_shannon[caractere] 10 return code

Exercice 2 (6 points)

Correction

Question	*Niveau*	*Contenu*	*Solution*
1	1	Base de données : Clé primaire	La valeur d'un de l'attribut `nom` n'est pas forcément unique.
2	1	Base de données : requête SQL	La requête SQL va afficher les noms et l'editeur correspondant de tous les jeux de la ludothèque, triés par ordre alphabétique.

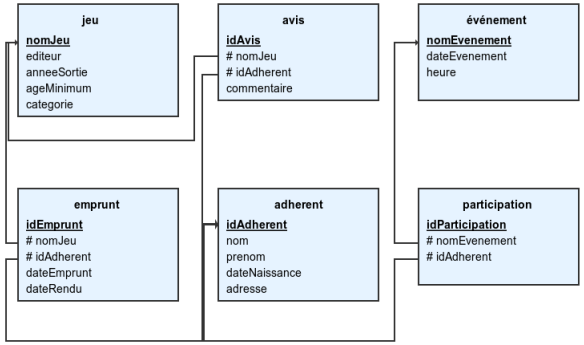
3	1	Base de données : requête SQL	`SELECT nomJeu`
			`FROM emprunt`
			`WHERE dateRendu = NULL;`

4	2	Base de données : requête SQL	`SELECT adherent.nom, adherent.prenom`
			`FROM adherent`
			`JOIN emprunt ON emprunt.adherent = adherent.idAdherent`
			`WHERE emprunt.nomJeu = "Catan";`

Question	*Niveau*	*Contenu*	*Solution*

5	1	Base de données : requête SQL	`UPDATE emprunt`
			`SET date_rendu = '2025-06-03'`
			`WHERE id_emprunt = 1538;`

6	1	Base de données : requête SQL	`SELECT nom, categorie`
			`FROM jeu`
			`WHERE annee_sortie >= 2010 AND age_minimum < 10;`

7	2	Base de données : schéma relationnel	 Figure 2 complétée.
8	3	Structures de	`1 dict_emprunts = {}` `2 for jeu in liste:`

Question	*Niveau*	*Contenu*	*Solution*
		données : dictionnaires	3 `    if jeu in dict_emprunts:` 4 `        dict_emprunts[jeu] += 1` 5 `    else:` 6 `        dict_emprunts[jeu] = 1`
9	3	Langages et programmation	1 `# exemple de solution` 2 `def le_podium(dico):` 3 `    podium = [[], [], []]` 4 `    scores_podium = [0, 0, 0]` 5 6 `    for i in range(3):` 7 `        for jeu, score in dico.items():` 8 `            if score > scores_podium[2-i]  \` 9 `            and score not in scores_podium:` 10 `                print(score)` 11 `                scores_podium[2-i] = score` 12 13 `    for i in range(3):` 14 `        for jeu, score in dico.items():` 15 `            if score==scores_podium[i]:` 16 `                podium[i].append(jeu)` 17 18 `    return podium`

Exercice 3 (8 points)

Correction

Question	*Niveau*	*Contenu*	*Solution*
1	2	Sécurisation des communications	PGRDX

2	2	Constructions élémentaires - Tableau indexé	`def indice(L, element):`
			`  n = len(L)`
			`  for i in range(n):`
			`  if element == L[i]:`
			`  return i`

3	2	Constructions élémentaires - Tableau indexé	`def lettres_vers_indices(chaine):`
			`  liste_indices = []`
			`  for lettre in chaine:`
			`  indice_lettre = indice(alphabet, lettre)`
			`  liste_indices.append(indice_lettre)`
			`  return liste_indices`

4	2	Constructions élémentaires	`for k in range(n):`
			`  ind = indices_msg[k] + indices_cle[k]`
			`  if ind >= 26:`
			`  ind = ind - 26`
			`  indices_msg_chiffre.append(ind)`

Question	*Niveau*	*Contenu*	*Solution*
			`msg_chiffre = indices_vers_lettres(indices_msg_chiffre)` `return msg_chiffre`
5	1	Mise au point des programmes - Gestion des bugs	Comme la longueur de la clé est inférieure à la longueur du message à chiffrer, une exception `AssertionError` est levée lors de l'exécution de la ligne 2. On n'attend pas de précision sur le message d'erreur qui s'affiche.
6	2	Sécurisation des communications	On obtient le message BRAVO
7	3	Sécurisation des communications	Pour chaque caractère du message chiffré, on soustrait, à sa position dans l'alphabet, la position du caractère associé dans le masque. Si la valeur obtenue est strictement négative, on ajoute 26. On obtient finalement la position du caractère en clair dans l'alphabet.

8	2	Sécurisation des communications - Constructions	`indices_msg_dechiffre = []`
			`for k in range(n):`
			`    ind = indices_msg[k] - indices_cle[k]`
			`    if ind < 0:`
			`        ind = ind + 26`
			`    indices_msg_dechiffre.append(ind)`
			`msg_dechiffre =`

Question	*Niveau*	*Contenu*	*Solution*
		élémentaires	`indices_vers_lettres(indices_msg_dechiffre)` `return msg_dechiffre`
9	1	Sécurisation des communications	Dans un algorithme de chiffrement symétrique, la même clé secrète est utilisée pour chiffrer et déchiffrer des messages. Un algorithme de chiffrement asymétrique repose sur l'existence d'une paire de clés : la clé privée, qui doit être gardée secrète par son propriétaire, et la clé publique qui peut être communiquée à tous.
10	1	Sécurisation des communications	Bob utilise sa clé privée pour déchiffrer le message.
11	2	Sécurisation des communications	Une tierce personne peut intercepter la clé publique de Bob et le message envoyé par Alice. Il remplace ensuite le message envoyé par Alice par un nouveau message, également chiffré à l'aide de la clé publique de Bob. Bob reçoit alors un message corrompu, qu'il peut déchiffrer avec sa clé privée : ce message ne provient pas d'Alice mais il ne s'en rend pas compte.
12	2	Sécurisation des communications	Lors d'une requête HTTPS, une liaison sécurisée est établie grâce à un algorithme de chiffrement asymétrique. Cette première étape permet l'authentification du serveur (qui envoie un certificat, établi par un tiers de confiance, prouvant son identité) et l'envoi de sa clé publique au client. Dans un second temps, le client construit et transmet au serveur une clé secrète. Les chiffrements des messages s'effectuent ensuite en s'appuyant sur un algorithme de chiffrement symétrique dont la clé est la clé secrète maintenant partagée.

Question	*Niveau*	*Contenu*	*Solution*
13	1	Sécurisation des communications	L'utilisation d'un algorithme de chiffrement asymétrique pour sécuriser intégralement les communications nécessiterait de faire beaucoup trop de calculs, un tel algorithme serait trop gourmand en ressources.
14	1	Transmission de données dans un réseau - Architecture d'un réseau	L'affichage obtenu indique que les paquets n'ont pas été transmis au destinataire et qu'ils sont perdus. Le poste de travail de Bob fait partie du réseau local d'adresse 192.168.110.0/24. Marc s'est donc trompé dans l'adresse IP : le troisième octet n'est pas le bon, il aurait dû saisir ping 192.168.110.115 pour atteindre le poste de travail de Bob.
15	1	Écriture d'un entier positif dans une base $b \geqslant 2$	255.255.255.224
16	1	Écriture d'un entier positif dans une base $b \geqslant 2$	32
17	1	Écriture d'un entier	134 a pour représentation binaire 10000110.

Question	*Niveau*	*Contenu*	*Solution*
		positif dans une base b $\geqslant$ 2	

18	3	Transmission de données dans un réseau - Architecture d'un réseau	L'affichage obtenu indique que les paquets ont été correctement transmis : le poste de travail de Zoé fait donc partie du même sous-réseau que le poste qu'elle essaie de joindre. Avec le choix de l'administratrice système, il suffit d'appliquer le masque sur le dernier octet des adresses IP pour réussir à identifier les deux postes qui font partie du même sous-réseau (les trois premiers octets seront toujours identiques).
			
			En appliquant le masque de sous-réseau sur le dernier octet de l'adresse IP de Zoé, on obtient `10000000`.
			
			`   1 1 1 0 0 0 0 0 (224)`
			`ET 1 0 0 0 0 1 1 0 (134)`
			`----------------`
			`   1 0 0 0 0 0 0 0 (128)`
			
			On obtient le même résultat pour le dernier octet de l'adresse IP du poste de travail de Marc :
			
			`   1 1 1 0 0 0 0 0 (224)`
			`ET 1 0 0 1 1 0 0 1 (153)`
			`----------------`
			`   1 0 0 0 0 0 0 0 (128)`
			
			Pour le poste de travail de Bob, le résultat est donné dans l'énoncé : `01100000 (96)`.

Question	*Niveau*	*Contenu*	*Solution*
			Les postes de travail de Zoé et Marc font donc partie du même sous-réseau. On en déduit que Zoé a exécuté la commande n°2.