# Exercice 1 (6 points)

Correction

| Question | Niveau | Contenu | Solution |
| --- | --- | --- | --- |
| 1 | 1 | Protocole de routage | | Table de routage | | | | --- | --- | --- | | destination | prochain saut | distance | |  |  | 0 | |  |  | 1 | |  |  | 0 | |  |  | 0 | |  |  | 1 | |
| 2 | 1 | Protocole de routage | Pour sortir du maillage à destination d’internet, la requête doit traverser le routeur .  La table précedente donne une route |
| 3 | 1 | Protocole de routage | |  |  |  | | --- | --- | --- | | Table de routage | | | | destination | prochain saut | distance | |  |  | 10 | |  |  | 11 | |  |  | 12 | |  |  | 13 | |  |  | 14 | |
| 4 | 1 | Protocole de routage | Pour sortir du maillage à destination d’internet, la requête doit traverser le routeur .  La table précedente donne une route |
| 5 | 1 | Protocole de routage | Pour sortir du maillage à destination d’internet, la requête doit toujours traverser le routeur .  La table précedente donne une route distance=101 |
| 6 | 1 | Première : binaire | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Calcul d’une adresse de réseau | | | | | | machine (binaire) | 11000000 | 10101000 | 00000001 | 01100100 | | masque (binaire) | 11111111 | 11000000 | 00000000 | 00000000 | | réseau (binaire) | 11000000 | 10000000 | 00000000 | 00000000 | | réseau (déc. pointée) | 192 | 128 | 0 | 0 | |
| 7 | 1 | Première : binaire | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Calcul d’une adresse de broadcast | | | | | | réseau (binaire) | 11000000 | 10000000 | 00000000 | 00000000 | | masque (binaire) | 11111111 | 11000000 | 00000000 | 00000000 | | complément. (binaire) | 00000000 | 00111111 | 11111111 | 11111111 | | broadcast (binaire) | 11000000 | 10111111 | 11111111 | 11111111 | | broadcast (déc. pointée) | 192 | 191 | 255 | 255 | |
| 8 | 1 | Syst. Exploitation et Réseaux | * adresse réseau : 172.16.0.0/16 * broadcast : 172.16.255.255 * disponibles, au maximum : adresses |
| 9 | 2 | Syst. Exploitation et Réseaux | 1 def masquer(self, masque: str)->str:  2 """  3 Détermine le préfixe masqué de l'adresse,  4 le masque (décimal pointé) étant passé en  5 paramètre.  6 >>> add = IPv4('192.168.1.100')  7 >>> add.masquer('255.192.0.0')  8 '192.128.0.0'  9 """ 10 tmp = [] 11 ip = self.octets() 12 crible = IPv4(masque).octets() 13 for i in range(4): 14 # Opération booléenne : 15 tmp.append(ip[i] & crible[i])  16 return ".".join([str(element) for element in tmp]) |
| 10 | 3 | Syst. Exploitation et Réseaux | 1 def adresse\_suivante(self, adresse\_max:str)->str:  2 """  3 Détermine l'adresse décimale pointée suivant   4 immédiatement l'adresse courante, sous réserve   5 d'existence d'une adresse disponible  6 >>> add = IPv4('192.168.1.100')  7 >>> add.adresse\_suivante('192.168.1.254')  8 '192.168.1.101'   9 >>> add = IPv4('192.168.1.255') 10 >>> add.adresse\_suivante('192.168.255.254') 11 '192.168.2.0' 12 """ 13 assert self.adresse < adresse\_max 14 liste\_courante = self.octets() 15 liste\_suivante = list() 16 retenue = 1 17 for index in range(4): 18 somme = liste\_courante[3 - index] + retenue 19 valeur, retenue = somme%256, somme//256  20 liste\_suivante = [str(valeur)] + liste\_suivante  21 return '.'.join(liste\_suivante) |

# Exercice 2 (6 points)

Correction

| Question | Niveau | Contenu | Solution |
| --- | --- | --- | --- |
| 1 | 1 | dévolution du problème | La première possibilité est de jouer la première case en plaçant un pion.    La seconde possibilité est de jouer la troisième case en enlevant le pion. |
| 2 | 1 | écriture d'une fonction | 1 def initialiser(n): 2 tab = [] 3 for i in range(n): 4 tab.append(False) 5 return tab  ou toute autre variante acceptée. |
| 3 | 1 | Compléter un programme | def victoire(tab):  for etat\_case in tab:  if etat\_case == False:   return False   return True |
| 4 | 2 | écriture d'une fonction avec parcours | 1 def indice\_premiere\_case\_occupee(tab): 2 for i in range(len(tab)): 3 if tab[i]: 4 return i 5 return None |
| 5 | 2 | coder une fonction | 1 def coup\_valide(tab, case): 2 if case==0: 3 return True 4 elif case==indice\_premiere\_case\_occupee(tab)+1 and case<len(tab) and case>=0: 5 return True 6 else: 7 return False |
| 6 | 1 | écriture d'une fonction | 1 def changer\_case(tab, case): 2 if coup\_valide(tab, case): 3 tab[case] = not tab[case] 4 return tab |
| 7 | 1 | concaténation | 1 def vider(n): 2 if n == 1: 3 print('Vider case 1')  4 elif n > 1: 5 vider(n-2) 6 print('Vider case '+str(n))  7 remplir(n-2) 8 vider(n-1) |
| 8 | 2 | détermination affichage d'une fonction récursive | Vider case 1  Vider case 3  Remplir case 1  Vider case 2  Vider case 1 |
| 9 | 3 | écriture d'une fonction récursive | 1 def remplir(n): 2 if n == 1: 3 print('Remplir case 1') 4 elif n > 1: 5 remplir(n-1) 6 vider(n-2) 7 print(f'Remplir case {n}') 8 remplir(n-2) |
| 10 | 2 | problème posé par une fonction récursive avec un baguenaudier de grande taille | En raison du grand nombre d’appels récursifs, la limite de taille de la pile des appels récursifs risque d’être dépassée ce qui provoquerait l’arrêt du programme avec une erreur de type RecursionError (ou tout autre justification cohérente). |

# Exercice 3 (8 points)

Correction

| Question | Niveau | Contenu | Solution |
| --- | --- | --- | --- |
| 1 | 1 | Programmation | gen\_mdp(10, True, True, False) # Le paramètre longueur doit être supérieur ou égal à 8. |
| 2 | 2 | Tableau donné en compréhension. | 8 minuscules = [chr(i) for i in range(97, 123)]   9 majuscules = [chr(i) for i in range(65, 91)]  10 caracteres\_speciaux = [chr(i) for i in range(33, 48)] + [chr(i) for i in range(58, 65)] |
| 3 | 2 | Programmation | 11 jeu\_caracteres = [] 12 if cont\_min: 13 jeu\_caracteres += minuscules  14 if cont\_maj:  15 jeu\_caracteres += majuscules  16 if cont\_spe:  17 jeu\_caracteres += caracteres\_speciaux |
| 4 | 1 | Programmation | mot\_de\_passe = mot\_de\_passe + jeu\_caracteres[randint(0, n-1)] |
| 5 | 2 | Mise au point des programmes. Gestion des bugs. | L’utilisation de randint à la ligne 21 choisit au hasard des caractères dans jeu\_caracteres pour créer le mot de passe. Il se peut donc que celui-ci ne contienne, par exemple, que des minuscules et aucun caractère spécial. |
| 6 | 1 | Modèle relationnel | L’attribut mot\_de\_passe est une clé primaire de la table compte : chaque valeur de cet attribut est donc nécessairement unique. Alice ne peut donc pas avoir deux fois le même mot de passe. |
| 7 | 1 | Langage SQL : requête d’interrogation. | SELECT url  FROM site; |
| 8 | 1 | Langage SQL : requête de mise à jour. | UPDATE compte  SET mot\_de\_passe = 'yhTS?d@UTJe'  WHERE mot\_de\_passe = '@rDfohpj!&'; |
| 9 | 1 | Langage SQL : requête d’interrogation. | SELECT site\_id  FROM compte  WHERE renouvellemement < '2024-03-20';  ou la réponse comportant un signe <=. |
| 10 | 1 | Mise au point des programmes. | Le format AAAA-MM-JJ permet plus facilement d’écrire des requêtes ordonnant les résultats par date de renouvellement du mot de passe. En effet, l’ordre lexicographique sur les chaînes au format AAAA-MM-JJ correspond à l’ordre chronologique, ce qui n’est pas le cas sur les chaînes au format JJ-MM-AAAA, où l’on a par exemple '10-10-2010' < '11-06-2010', alors que le 10 octobre est postérieur au 11 juin de la même année. |
| 11 | 3 | Langage SQL : requête d’interrogation. | SELECT utilisateur, mot\_de\_passe  FROM compte  JOIN site ON compte.id\_site = site.id  WHERE nom\_site = 'Votremailp'  ORDER BY renouvellement; |
| 12 | 2 | Base de données relationnelle (repérer des anomalies) | La solution à une seule table impliquerait la redondance des informations (nom\_site et url) dans le cas où Alice possède plusieurs comptes chez un même service.  ou toute autre réponse cohérente. |
| 13 | 2 | Systèmes d’exploitation. Programmation | chiffrement('gestionnaire.db','../Perso/secret.db','../Perso/cle')  ou également des chemins absolus. |
| 14 | 2 | Écriture d’un entier positif dans une base | La représentation binaire de A3 est 10100011. La représentation binaire de 59 est 01011001. 10100011 XOR 01011001 = 11111010  Le résultat exprimé en binaire (11111010), décimal (250) ou hexadécimal (FA). |
| 15 | 1 | Expressions booléennes | Le candidat pourra, par exemple, établir la table de vérité de (a XOR b) XOR b en fonction de a et b et montrer que les colonnes (a XOR b) XOR b et a sont identiques |
| 16 | 3 | Sécurisation des communications. | Le chiffrement est symétrique. En effet, si a est le bit à chiffrer et b le bit de clé, on a alors a XOR b qui corespond au bit chiffré. La propriété montre qu’en appliquant le même bit de clé b sur le bit chiffré, on retrouve le bit initial a. La même clé b a donc permis de chiffrer et de déchiffrer. |
| 17 | 2 | Utiliser les commandes de base en ligne de commande. Gérer les droits et permissions d’accès aux fichiers. | Tous les utilisateurs utilisant le système d’Alice possèdent les droits de lecture sur le fichier secret.db. Ils peuvent donc notamment ouvrir ce fichier et tenter une attaque. Alice peut corriger ce problème en modifiant les permissions sur le fichier (par exemple avec la commande chmod, dont le nom n’est pas attendu). |
| 18 | 3 | Mise au point des programmes. Gestion des bugs. | * Le mot de passe différent pour chaque site est respecté par Alice. En effet, elle a choisi mot\_de\_passe comme clé primaire de la table compte, ce qui impose l’unicité des mots de passe ; * Création de mots de passe suffisamment longs, complexes et inattendus : oui, dans la mesure où Alice effectue des appels à la fonction gen\_mdp avec un paramètre longueur suffisamment grand, idéalement bien supérieur à 8. On peut aussi remarquer que la règle énoncée peut porter à confusion sur le fait que le mot de passe PEUT ou DOIT contenir tel ou tel type de caractère. Idéalement, il ne faut pas imposer que le mot de passe contienne tel ou tel type de caractères pour diminuer le nombre d’informations connues sur le mot de passe ; * Non communication des mots de passe à un tiers : cette bonne pratique est respectée par le chiffrement de gestionnaire.db et par l’application de permissions adaptées ; * Utilisation d’un gestionnaire de mots de passe : oui, c’est le projet d’Alice. |