Exercice 1 (6 points)

Correction

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
1		1	Manupulation de tableaux	E P R U V D N S I A B C F G H J K L M O Q T X Y Z
2		1	Manupulation de tableaux	`assert 'W' not in clef`

3		1	Assertions	`def creer_carre(liste_clef):`
				`  carre = [[0 for i in range(5)] for j in range(5)]`
				`  for i in range(25):`
				`  carre[i//5][i%5] = liste_clef[i]`
				`  return carre`

4		1	couper_en_digrammes	`def couper_en_digrammes(message):`
				`  digrammes = []`
				`  i = 0`
				`  while i < len(message) - 1:`
				`  if message[i] == message[i+1]:`
				`  digrammes.append(message[i] + 'X')`
				`  i = i + 1`
				`  else:`
				`  digrammes.append(message[i] +`

Question	*Barème*	*Niveau*	*Contenu*	*Solution*

				`message[i+1])`
				`10 i = i + 2`
				
				`11 if i == len(message) - 1: #il reste une lettre isolée`
				`12 digrammes.append(message[i] + 'X')`
				`13 return digrammes`

5		1	Programmation	`["BO","NJ","OU","RX"]`

6		2	Algorithme	`BO -> HV`
				`NJ -> QG`
				`OU -> NV`
				`RX -> CU`
				Soit le chiffré `HVQGNVCU`

7		1	Manupulation de tableaux	`def ligne_colonne(lettre, carre):`
				`  for i in range(len(carre)):`
				`  for j in range(len(carre)):`
				`  if carre[i][j] == lettre:`
				`  return (i,j)`

8		1	Manupulation de tableaux	`def sur_la_meme_ligne(digramme, carre):`
				`  lettre1 = digramme[0]`
				`  lettre2 = digramme[1]`

Question	Barème	Niveau	Contenu	Solution
				`4 return ligne_colonne(lettre1, carre)[0]==ligne_colonne(lettre2, carre)[0]`

9		3	Programmation	`1 def chiffrer_digramme(digramme, carre):`
				`2 lettre1 = digramme[0]`
				`3 lettre2 = digramme[1]`
				`4 i1, j1 = ligne_colonne(lettre1, carre)`
				`5 i2, j2 = ligne_colonne(lettre2, carre)`
				`6 if sur_la_meme_ligne(digramme, carre):`
				`7 digramme_chiffre = carre[i1][(j1 + 1)%5] + carre[i2][(j2 + 1)%5]`
				`8 elif sur_la_meme_colonne(digramme, carre):`
				`9 digramme_chiffre = carre[(i1 + 1)%5][j1] + carre[(i2 + 1)%5][j2]`
				`10 else:`
				`11 digramme_chiffre = carre[i1][j2] + carre[i2][j1]`
				`12 return digramme_chiffre`

10		3	Programmation	`1 def chiffrer_playfair(message, clef):`
				`2 carre = creer_carre(creer_liste_clef(clef))`
				`3 d_msg = couper_en_digrammes(message)`
				`4 msg_chiffre = ""`
				`5 for d in d_msg:`
				`6 msg_chiffre += chiffrer_digramme(d, carre)`
				`7 return msg_chiffre`

Exercice 2 (6 points)

Correction

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
1		1	Utiliser les commandes de base en ligne de commande	La commande `ls` permet d'obtenir le listing des fichiers et dossiers présents. On ne demande pas l'option `-l`.
2		1	Gérer les droits et permissions d'accès aux fichiers	`-rwxrw-r--` signifie que le propriétaire peut lire, écrire/modifier et exécuter le fichier. les autres membres du groupe peuvent lire et écrire/modifier les autres utilisateurs ne peuvent que le lire.
3		1	Utiliser les commandes de base en ligne de commande	`rm concat.pls`. On accepte `rm` seul.
4		1	Utiliser la documentation d'une bibliothèqu	`chmod g-wx` indique que les permissions en écriture et en exécution sont retirées aux membres du groupe `adminsys`.

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
			e, d'une command e.	
5		1	Passer de la représenta tion d'une base dans une autre.	`rw-` donne 110 qui vaut 6

6		2	Programm ation algorithmiq ue	`def bin_to_oct(chaine):`
				`  valeur = 0`
				`  for i in range(len(chaine)):`
				`  valeur = valeur + int(chaine[i]) * 2 ** (2 - i)`
				`  return valeur`
				
				ou
				
				`def bin_to_oct(chaine):`
				`  valeur = 0`
				`  poids = 1`
				`  for i in range(len(chaine)-1, -1, -1):`
				`  valeur = valeur + int(chaine[i]) * poids`
				`  poids = poids * 2`
				`  return valeur`
				
				ou tout autre réponse valide.

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
7		2	Algorithmique	On obtient en sortie la chaîne `'110101100'`.

8		2	Programmation algorithmique	`def symb_to_oct(chaine):`
				`  repr_bin = symb_to_bin(chaine)`
				`  # les 3 premiers bits correspondent au propriétaire`
				`  user = repr_bin[0] + repr_bin[1] + repr_bin[2]`
				`  # les 3 suivants au groupe`
				`  group = repr_bin[3] + repr_bin[4] + repr_bin[5]`
				`  # et les 3 derniers aux autres`
				`  other = repr_bin[6] + repr_bin[7] + repr_bin[8]`
				`  return str(bin_to_oct(user)) + str(bin_to_oct(group)) + str(bin_to_oct(other))`
				ou tout autre réponse valide en une ou plusieurs lignes.

9		1	Vocabulaire et utilisation de la programmation objet.	`mon_fichier = Fichier('concat.pls', 1257, 'root', 'www-data', 'rw-r--r--')`

10		2	Programmation objet	on attend une méthode
				`def chown(self, nouveau_proprietaire):`
				`  self.proprietaire = nouveau_proprietaire`

11		2	Programmation objet	`def get_executable(fichiers, proprietaire):` `  executables = []`

Question	Barème	Niveau	Contenu	Solution
				3 for fichier in fichiers: 4 if fichier.proprietaire == proprietaire and fichier.permission[2] == 'x': 5 executables.append(fichier) 6 return executables

Exercice 3 (8 points)

Correction

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
1		1	BDD insertion	`INSERT INTO Produits` `VALUES (10,'Croissant', 'Alimentaire', 0.90, 4)`

2		1	BDD mise à jour	`UPDATE Fournisseurs`
				`SET adresse = '78 Rue des Jeux', ville = 'Elbeuf',`
				`WHERE nom_fournisseur = 'Livres en Folie';`

3		1	BDD comprendre requête SELECT FROM WHERE	Cette requête SQL sélectionne et affiche les noms de tous les produits de la catégorie `Alimentaire`. nom --- Yaourts blanc x 4 Lait Pain On accepte la réponse sous la forme d'une phrase ou d'un tableau.

4		1	BDD requête avec 1 condition	`SELECT *`
				`FROM Commandes`
				`WHERE total >= 1000;`

Question	*Barème*	*Niveau*	*Contenu*	*Solution*

5		1	BDD requête condition double OR	`SELECT nom`
				`FROM Fournisseurs`
				`WHERE pays ='Espagne' OR pays='Italie' ;`

6		2	BDD requête jointure	`SELECT Fournisseurs.nom`
				`FROM Fournisseurs`
				`JOIN Produits ON Fournisseurs.id_fournisseur = Produits.id_fournisseur`
				`WHERE catégorie = 'Alimentaire';`

7		3	requête 3 jointures	`SELECT id_commande, date, Fournisseurs.nom`
				`FROM Fournisseurs`
				`JOIN Commandes ON Fournisseurs.id_fournisseur = Commandes.id_fournisseur`
				`JOIN Details ON Commandes.id_commande = Details.id_commande`
				`JOIN Produits ON Details.id_produit = Produits.id_produit`
				`WHERE categorie = 'Vêtement';`

8		1	Graphe chemin & algo de Dijkstra	Le plus court chemin est : Toulouse - Bordeaux - Nantes - Calais

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
9		2	Graphe Parcours profondeur	Calais – Nantes – Bordeaux – Toulouse – Lyon – Marseille - Paris – Strasbourg
10		2	Graphe Parcours profondeur	Calais – Nantes – Paris – Strasbourg – Bordeaux – Lyon – Toulouse – Marseille
11		1	ajout dans un dictionnaire	`graphe['Lyon']['Marseille'] = 315` `graphe['Marseille']['Lyon'] = 315`

12		2	parcours dico	`def distance(graphe, ville1, ville2):`
				`  if ville1 in graphe and ville2 in graphe[ville1]:`
				`  return graphe[ville1][ville2]`
				`  else:`
				`  return None`

13		3	calcul d'une somme avec parcours d'un cido	`def distance_totale(graphe, ville_depart, ville_destination):`
				`  chemin = trouver_chemin(graphe, ville_depart, ville_destination)`
				`  if chemin is None:`
				`  return None`
				
				`  distance_totale = 0`
				`  for i in range(len(chemin) - 1):`
				`  ville_courante = chemin[i]`

Question	*Barème*	*Niveau*	*Contenu*	*Solution*
				9 prochaine_ville = chemin[i + 1] 10 distance_totale += graphe[ville_courante][prochaine_ville] 11 return distance_totale
14		1	accès à une info dans une liste qui est une valeur d'un dico	La valeur de `graphe['Paris']['Nantes'][1]` est `3.47`.
15		2	compléter code - utilisation dico	1 def ratio_duree_distance(graphe): 2 for ville, connexions in graphe.items(): 3 for destination, valeurs in connexions.items(): 4 distance, duree = valeurs 5 ratio = duree / distance 6 graphe[ville][destination].append(ratio) 7 return graphe
16		1	comprendre un algo -> algo glouton	Algorithme glouton.
17		2	algo glouton	`['Toulouse', 'Bordeaux', 'Nantes', 'Paris', 'Calais']`