Exercice 1 (6 points)

Correction

Question	Barème	Niveau	Contenu	Solution
1		1	mobiliser une définition	C'est ABCEDF car dans un parcours en largeur les voisins directs du sommet de départ doivent être visités avant les autres.
2		1	appliquer un protocole à la main	Table de routage de F routeur | nombre de sauts | prochain routeur D | 1 | –
3		1	appliquer un protocole à la main	Table de routage de A routeur | nombre de sauts | prochain routeur B | 1 | – C | 1 | – E | 1 | – D | 2 | C ou E
4		1	itérer un protocole	C'est à l'itération n°3 que plus aucune table ne change.
5		1	itérer un protocole	Table de routage de A routeur | nombre de sauts | prochain routeur

Question	Barème	Niveau	Contenu	Solution
				B 1 – C 1 – E 1 – D 2 C ou E F 2 E

6		1	compléter une fonction Python	`if autre not in self.voisins:`
				`  self.voisins.append(autre)`
				`  self.nb_sauts[autre] = 1`
				`  self.prochain[autre] = None`
				`  if self not in autre.voisins:`
				`  autre.relie(self)`

7		1	produire entièrement une fonction Python simple	`def relie_liste(self, lst):`
				`  for routeur in lst:`
				`  self.relie(routeur)`

8		1	programmer une situation simple	`C.relie_liste([E, D])`
				`D.relie_liste([E, F])`
				D'autres solutions sont possibles.

Question	Barème	Niveau	Contenu	Solution

9		2	compléter une fonction Python	`for r in autre.nb_sauts:`
				`  if r != self:`
				`  if (r not in self.nb_sauts or`
				`  self.nb_sauts[r] > autre.nb_sauts[r] + 1):`
				`  self.nb_sauts[r] = autre.nb_sauts[r] + 1`
				`  self.prochain[r] = autre`

10		1	écrire une fonction Python simple	`  def itere_rip(self):`
				`  for v in self.voisins:`
				`  self.met_a_jour_table(v)`

11		1	écrire un code Python simple	`for routeur in liste_routeurs:` `  routeur.itere_rip()`

12		3	imaginer/produire des modifications à un programme Python complexe	`def itere_rip(self):`
				`  modif = False`
				`  for v in self.voisins:`
				`  b = self.met_a_jour_table(v)`
				`  modif = b or modif`
				`  return modif`

13		3	imaginer/produire	`mise_a_jourp = True` `while mise_a_jourp:`

Question	Barème	Niveau	Contenu	Solution

			des modifications à un programme Python complexe	`    new_infop = False`
				`    for routeur in liste_routeurs:`
				`        b = routeur.itere_rip()`
				`        new_infop = b or new_infop`
				`    mise_a_jourp = new_infop`

Exercice 2 (6 points)

Correction

1		1	lecture d'une requete simple	Lac Vert
2		1	lecture de tables	Ils peuvent aller au Lac Blanc et aux Lacs Noirs.

3		1	requete simple	`SELECT coord_GPS`
				`FROM parking`
				`WHERE commune = 'Passy'`
				`AND altitude > 800`
				`AND altitude < 1000;`

4		2	requete complexe	`SELECT lac.nom`
				`FROM lac`
				`JOIN rando ON rando.arrivee = lac.idL`
				`JOIN parking ON parking.idP = rando.depart`
				`WHERE parking.altitude = 1300 AND parking.commune = 'Cordon';`

5		2	clé étrangère	Il faut commencer par insérer le lac avant la randonnée.
				`INSERT INTO lac VALUES (42, 'Lac d Anterne', 2059);`
				`INSERT INTO rando VALUES (100, 3, 42);`

6		1	requete simple	`UPDATE lac`
				`SET nom = 'Lac d Anterne'`
				`WHERE nom = 'Lc d Anterne';`

7		2	requetes simples	Il faut commencer par supprimer les randonnées qui utilisent ce parking.
				`DELETE FROM rando`
				`WHERE depart = 28;`
				`DELETE FROM parking`
				`WHERE idP = 28;`

8		2	algo simple avec POO	1 def get_parking(randos): 2 parkings = [] 3 for rando in randos: 4 if rando.depart not in parkings: 5 parkings.append(rando.depart) 6 return parkings
9		1	algo simple	1 def get_nb_rando(parking, randos): 2 nb = 0 3 for rando in randos: 4 if rando.depart == parking: 5 nb = nb + 1 6 return nb
10		3	boucle avec utilisation de fonction	1 def nb_rando_par_parking(randos): 2 parkings = get_parking(randos) 3 4 nb_rando = {} 5 for parking in parkings: 6 nb_rando[parking] = get_nb_rando(parking, randos) 7 return nb_rando

Exercice 3 (8 points)

Correction

1		1	Modéliser un problème	La valeur minimale d'une cible est 1 ce qui correspond à une seule case égale à 1. La valeur maximale d'une cible est 45 ce qui correspond à toutes les valeurs égales à 9.
2		1	Modéliser un problème	La ligne *L3* est composée des chiffres 6, 4, 5, 8, 2. Ainsi sa cible minimale est 2 (le minimum de la ligne). Sa cible maximale est 6+4+5+8+2 = 25.
3		1	Coder en Python	`def extraireLigne(plateau, indiceLigne):` `  return plateau[indiceLigne]`

4		2	Comprendre une représentation, traduire un algorithme dans un langage de programmation	`def extraireColonne(plateau, indiceColonne):`
				`  return [plateau[i][indiceColonne] for i in range(5)]`
				`# ou par exemple`
				`def extraireColonne(plateau, indiceColonne):`
				`  colonne = []`
				`  for i in range(5):`
				`  colonne.append(plateau[i][indiceColonne]`
				`  return colonne`

5		1	Coder en Python	`solution = [[0, 9, 2, 0, 2],`
				`[8, 0, 0, 0, 1],`
				`[7, 0, 3, 2, 0],`
				`[0, 4, 0, 0, 2],`
				`[0, 0, 0, 0, 4]]`

6		1	Comprendre un algorithme	`[[7, 9, 2, 0, 2],`
				`[8, 6, 3, 0, 1],`
				`[7, 7, 3, 2, 7],`
				`[6, 4, 5, 0, 2],`
				`[0, 0, 0, 0, 4]]`

7		2	Comprendre et compléter un code en Python pour traduire un algorithme.	`def regle1(plateau, ciblesLignes, ciblesColonnes):`
				`    for i in range(5):`
				`        tab = extraireLigne(plateau, i)`
				`        cible = ciblesLignes[i]`
				`        for j in range(5):`
				`            if tab[j] > cible:`
				`                plateau[i][j] = 0`
				`    for j in range(5):`
				`        tab = extraireColonne(plateau, j)`
				`        cible = ciblesColonnes[j]`
				`        for i in range(5):`
				`            if tab[i] > cible:`
				`                plateau[i][j] = 0`

8		2	Coder en Python	`def unImpair(tab):`
				`    n = 0`
				`    for el in tab:`
				`        if el % 2 == 1:`

				5 n = n + 1 6 return n == 1
9		2	Comprendre et compléter un code en Python pour traduire un algorithme.	1 def regle2(plateau, ciblesLignes, ciblesColonnes): 2 for i in range(5): 3 ligne = extraireLigne(plateau, i) 4 cible = ciblesLignes[i] 5 if cible % 2 == 0 and unImpair(ligne): 6 for j in range(5): 7 if plateau[i][j] % 2 == 1: 8 plateau[i][j] = 0 9 for j in range(5): 10 colonne = extraireColonne(plateau, j) 11 cible = ciblesColonnes[j] 12 if cible % 2 == 0 and unImpair(colonne): 13 for i in range(5): 14 if plateau[i][j] % 2 == 1: 15 plateau[i][j] = 0
10		1	Analyser une situation.	La liste `[1, 1, 0, 0, 1]` est un masque solution du problème proposé car 1+2+2=5. Les autres masques solutions sont `[0, 1, 1, 0, 0]`, `[0, 0, 1, 0, 1]`, `[0, 0, 0, 1, 0]`.
11		3	Concevoir un algorithme	1 def somme(tab, masque): 2 s = 0 3 for i in range(5): 4 if masque[i] == 1:

				5 s = s + tab[i] 6 return s
12		1	Représentation des entiers positifs en binaire	26 est représenté par la liste `[1, 1, 0, 1, 0]`.
13		1	Représentation des entiers positifs en binaire	Sur 5 bits, le minimum est `[0, 0, 0, 0, 0]`, soit 0 en décimal. Le maximum est `[1, 1, 1, 1, 1]`, soit 31 en décimal.
14		3	Représentation des entiers positifs en binaire, écrire un code complexe sans rien.	1 def dec2bin(n): 2 sortie = [0 for i in range(5)] 3 i = 4 4 while n > 0: 5 sortie[i] = n%2 6 n = n//2 7 i = i-1 8 return sortie 9 # ou par exemple 10 def dec2bin(entier): 11 representation = [0 for i in range(5)] 12 for i in range(4, -1, -1): 13 representation[i] = entier%2

				14 entier = entier//2 15 return representation
15		3	Concevoir un algorithme	1 def masques_solutions(tab, cible): 2 solutions = [] 3 for i in range(32): 4 tab_bin = dec2bin(i) 5 if somme(tab, tab_bin) == cible: 6 solutions.append(tab_bin) 7 return solutions
16		3	Concevoir un algorithme	1 def teste_solution(tab, ciblesL, ciblesC): 2 for i in range(5): 3 somme = 0 4 ligne = extraireLigne(tab,i) 5 for case in ligne: 6 somme = somme+case 7 if ciblesL[i] != somme: 8 return False 9 for j in range(5): 10 somme = 0 11 colonne = extraireColonne(tab,j) 12 for case in colonne: 13 somme = somme+case 14 if ciblesC[j] != somme: 15 return False 16 return True