Exercice 1 (6 points)

Correction

Q	N	Thème	Réponse
1	1	Arbre binaire	Oui on peut l'identifier, il s'agit d'un sorbier.
2	1	Arbre binaire	Non on ne peut pas l'identifier, aucun végétal connu n'a ces caractéristiques.

3	1	Programmation Orientée Objet	`sorbier = Feuille_resultat(["Sorbier"])`
			`robinier_noyer = Feuille_resultat(["Robinier", "Noyer"])`
			`feuille_vide1 = Feuille_resultat([])`
			`feuille_vide2 = Feuille_resultat([])`
			`n_bord = Noeud("Bord denté ?", sorbier, robinier_noyer)`
			`n_alt = Noeud("Alternées ?", n_bord, feuille_vide1)`
			`arbre_2 = Noeud("Simples ?", feuille_vide2, n_alt)`

4	1	Programmation Orientée Objet	Pour la classe `Noeud` :
			`def est_resultat(self):`
			`  return False`

5	1	Programmation Orientée Objet	Pour la classe `Feuille_resultat` :
			`def est_resultat(self):`
			`  return True`

Q	N	Thème	Réponse

6	1	Programmation Orientée Objet	Pour la classe `Feuille_resultat` :
			`1 def nb_vegetaux(self):`
			`2 return len(self.vegetaux)`

7	2	Algorithmes sur les arbres binaires	Pour la classe `Feuille_resultat` :
			`1 def nb_vegetaux(self):`
			`2 return self.sioui.nb_vegetaux() + self.sinon.nb_vegetaux()`

8	1	Programmation Orientée Objet	Pour la classe `Feuille_resultat` :
			`1 def liste_questions(self):`
			`2 return []`

9	2	Algorithmes sur les arbres binaires	Pour la classe `Feuille_resultat` :
			`1 def liste_questions (self):`
			`2 return [self.question] + self.sioui.liste_questions() + self.sinon.liste_questions ()`

10	2	Algorithmes sur les arbres binaires	`1 def est_bien_renseigne (dico_vegetal, arbre) :`
			`2 liste_q = arbre.liste_questions()`
			`3 for q in liste_q :`
			`4 if not q in dico_vegetal :`

Q	N	Thème	Réponse
			5 return False 6 return True
11	3	Algorithmes sur les arbres binaires	1 def identifier_vegetaux(arbre, dico_vegetal): 2 if arbre.est_resultat(): 3 return arbre.vegetaux 4 if dico_vegetal[arbre.question]: 5 return identifier_vegetaux(arbre.sioui, dico_vegetal) 6 else: 7 return identifier_vegetaux(arbre.sinon, dico_vegetal)

Exercice 2 (6 points)

Correction

Q	N	Thème	Réponse
1	1	Programmation Python, POO	`def passer_transit(self):` `    self.etat = 'transit'`

2	2	Programmation Python	`def ajouter_colis(liste, colis):`
			`    if colis.poids <= 25:`
			`        liste.append(colis)`
			`    else:`
			`        print('Dépassement du poids maximal autorisé')`

3	1	Parcours séquentiel d'un tableau	`def nb_colis(liste):` `    return len(liste)`

4	2	Parcours séquentiel d'un tableau	`def poids_total(liste):`
			`    total = 0`
			`    for c in liste :`
			`        total = total + c.poids`
			`    return total`

Q	N	Thème	Réponse

5	2	Parcours séquentiel d'un tableau, POO	`def liste_colis_etat(liste, statut):`
			`    resultat = []`
			`    for c in liste:`
			`        if c.etat == statut:`
			`            resultat.append(c)`
			`    return resultat`

6	1	Tris par sélection	Le tri par sélection de coût quadratique.
7	1	Tris	Le tri par insertion (coût quadratique) ou le tri fusion (coût *n log n*), ou n'importe quel autre tri.

8	3	Algorithme glouton, récursivité	`def chargement_glouton(liste, rang, capacite):`
			`    if rang == len(liste):`
			`        return []`
			`    elif liste[rang].poids <= capacite:`
			`        return [liste[rang]]  + chargement_glouton(liste, rang+1, capacite - liste[rang].poids)`
			`    else:`
			`        return chargement_glouton(liste, rang + 1, capacite)`

Q	N	Thème	Réponse
9	2	recursivité	La récursivité utilise une pile d'appels qui est un espace mémoire particulièrement limité, cela génère rapidement des débordements de capacité.

10	3	Algorithme glouton	`def chargement_glouton2(liste, capacite):`
			`    #initialisation des variables`
			`    poids_total = 0`
			`    colis_a_charger = []`
			`    # ajout si possible du prochain colis`
			`    for colis in liste_trie:`
			`        if poids_total + colis.poids <= capacite :`
			`            colis_a_charger.append(colis)`
			`            poids_total = poids_total + colis.poids`
			`    return colis_a_charger`

Exercice 3 (8 points)

Correction

Q	N	Thème	Réponse
1	1	bases_de_donnees	L'attribut `annee` devrait être un entier.
2	2	bases_de_donnees	Pour être cohérent, on pourrait contraindre ses valeurs de 2007 à 2025 qui correspondrait aux dates de naissances d'enfants (≤ 18).
3	1	bases_de_donnees	L'attribut `parent` en tant que clef étrangère suit une contrainte de référence.
4	1	bases_de_donnees	L'attribut `tel` permet d'identifier de façon unique un parent et peut donc être une clef primaire. (Un nom n'est pas forcément unique car il y a des homonymes, et le code postal n'est pas unique à une personne non plus).
5	2	bases_de_donnees	Cette requête lève une erreur car les enfants Maya (`17`) et Rachelle (`23`) ont pour parent le parent `33600781122`.

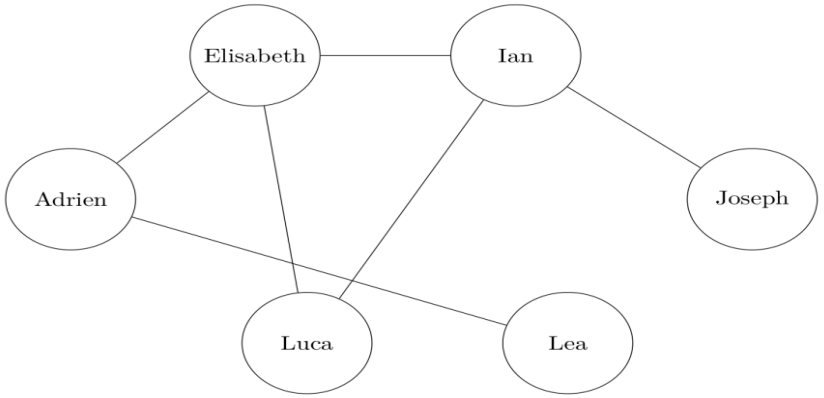
6	1	bases_de_donnees	`INSERT INTO parent VALUES ('Bauges', 33619782812, 73340);`
			`UPDATE enfant`
			`SET num_parent = 33619782812`
			`WHERE num_parent = 33600782812;`
			`DELETE FROM parent WHERE tel = 33600782812;`

Q	N	Thème	Réponse
7	1	bases_de_donnees	Le résultat est Nakamura, Hawa, Kian, Adrien

8	1	bases_de_donnees	`SELECT prenom`
			`FROM enfant`
			`WHERE num_parent = 33619861122`
			`ORDER BY prenom;`

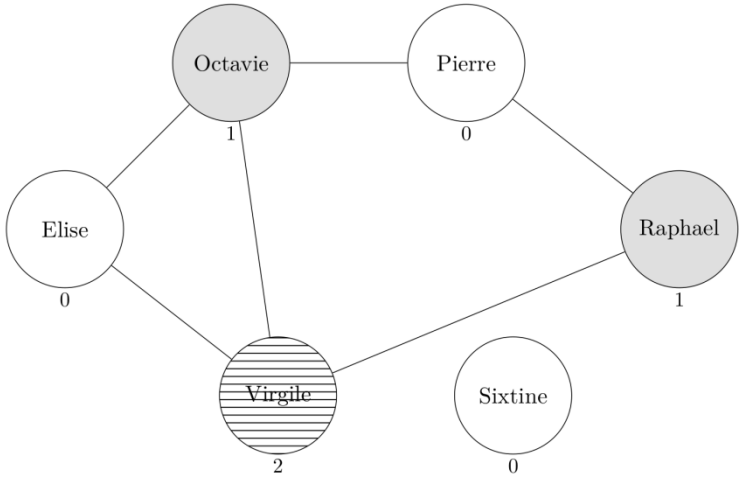
9	2	bases_de_donnees	`SELECT id, prenom`
			`FROM enfant JOIN parent ON num_parent = tel`
			`WHERE codep = 38520;`

10	1	graphe	La mésentente entre deux enfants est une relation symétrique, ainsi si l'enfant A est en mésentente avec l'enfant B il en sera de même de l'enfant B avec l'enfant A. Un graphe non-orienté suffit donc.

Q	N	Thème	Réponse
11	1	graphe	

12	1	langages_programmation	À l'aide d'une boucle ou bien
			`def degre(g, s):`
			`    return len(g[s])`

13	3	algorithmique	`for i in range(1, n):`
			`    sommet_courant = sommets[i]`
			`    j = i-1`
			`    while j >= 0 and degre(g, sommets[j]) < degre(g, sommet_courant):`
			`        sommets[j+1] = sommets[j]`
			`        j = j - 1`
			`    sommets[j+1] = sommet_courant`

Q	N	Thème	Réponse
14	2	algorithmique	Il s'agit d'un tri par insertion de coût d'exécution temporel quadratique.
15	2	graphe	

16	2	algorithmique	`def colorer_graphe(g, dc):`
			`    # Pré-condition : les clés de dc sont les sommets de g, et les valeurs de dc sont à -1`
			`    for s in dc:`
			`        couleur = plus_petite_couleur_hors_voisins(g, dc, s)`
			`        dc[s] = couleur`

Q	N	Thème	Réponse
17	3	algorithmique	1 def welsh_powell(g): 2 # initialisation à -1 pour tous les sommets dans le dictionnaire dc 3 dc = {s: -1 for s in g} 4 # coloration en suivant l'approche de Welsh-Powell 5 for s in sommets_tries(g): 6 couleur = plus_petite_couleur_hors_voisins(g, dc, s) 7 dc[s] = couleur 8 return dc