Exercice 1 — 6 points

| *Questions* | *Contenu et notions* | *Capacités exigibles* | *Éléments de réponses et commentaires* |
|---|---|---|---|
| 1 | programmation Python; POO | N1 | `self.jour = jour`<br>`self.mois = mois`<br>`self.annee = annee` |
| 2 | programmation Python; POO | N1 | 1 mai 2000 (01/05/2000) |
| 3 | programmation Python; POO | N1 | `d = Date(19, 6, 2024)` |
| 4 | programmation Python; POO | N1 | `def get_annee(self) -> int:`<br>`    return self.annee` |
| 5 | programmation Python; POO | N1 | `def set_mois(self, mois : int) -> None:`<br>`    self.mois = mois` |
| 6 | programmation Python ; POO | N2 | `if self.est_bissextile() :`<br>`    self.nb_jours_par_mois[1] = 29` |
| 7 | programmation Python ; POO | N3 | `def est_bissextile(self):`<br>`    an = self.annee`<br>`    return (an % 4 == 0 and an % 100 != 0) or an % 400 == 0:` |
| 8 | programmation Python ; POO | N1 | `>>> d1 = Date(20, 3, 2001)`<br>`>>> d1.nb_jours_passes()`<br>`79` |
| 9 | programmation Python ; POO | N3 | `def nb_jours_restants(self) -> int:`<br>`  j = 365`<br>`  if self.est_bissextile():`<br>`     j = 366`<br>`  return j - self.nb_jours_passes()` |

| Exercice 1 | 6 points | | |
|---|---|---|---|
| 10 | programmation<br>Python ; POO | N1 | `>>> d1.nb_jours_depuis(d2)`<br>`0`<br>`>>> d1.nb_jours_depuis(d3)`<br>`-1`<br>`>>> d1.nb_jours_depuis(d4)`<br>`-1`<br>`>>> d1.nb_jours_depuis(d5)`<br>`731` |
| 11 | programmation<br>Python ; POO | N2 | `def timestamp(self) -> int:`<br>`    d = Date(1, 1, 1970)`<br>`    return self.nb_jours_depuis(d) * 24 * 3600` |

| Exercice 2 | 6 points | | |
|---|---|---|---|
| 1 | Gestion de processus | N1 | Ordonnanceur |
| 2 | Gestion de processus | N1 | Elu, prêt, Bloqué (ou équivalent) |
| 3 | Structure de données | N1 | 2 File |
| 4 | Structure de données | N1 | `class Processus:`<br>`4 def __init__(self, PID, priorite, temps_CPU):`<br>`5  self.priorite=priorite`<br>`7  self.PID=PID`<br>`8  self.temps_utilisation=0`<br>`9  self.temps_CPU = temps_CPU` |
| 5 | Gestion de processus | N1 | `Cycle 1: CPU=P1 liste_files=[[P3, P2], [], []]`<br><br>`Cycle 2: CPU=P2 liste_files=[[P3],[P1],[]]`<br><br>`Cycle 3: CPU=P3 liste_files=[[], [P2, P1],[]]`<br><br>`Cycle 4: CPU=P3 liste_files=[[], [P2, P1], []]`<br><br>`Cycle 5: CPU=P1 liste_files=[[], [P2], [P3]]` |
| 6 | Gestion de processus | N2 | la priorité du processus lent va finir pas dépasser 4 et il n'aura plus jamais la priorité. |
| 7 | Gestion de processus | N2 | le temps d'attente va lui permettre de retrouver une priorité suffisante pour être élu. |
| 8 | liste de listes | N3 | `1 def meilleur_priorite(liste_files):`<br>`2  for priorite in range(len(liste_files)):`<br>`3  if len(file[priorite]) > 0:`<br>`4  return priorite`<br>`5  return None` |
| 9 | liste de listes | N3 | `1 def prioritaire(liste_files):`<br>`2  priorite = meilleur_priorite(liste_files)`<br>`3  if priorite == None:`<br>`4  return None`<br>`5  else:`<br>`6  return liste_files[priorite].pop(0)` |

| Exercice 2 | 6 points | | |
|---|---|---|---|
| 10 | liste de listes | N3 | |

```
def gerer(p, liste_files):
    if p == None or p.temps_execution >= p.temps_CPU:
        return prioritaire(liste_files)
    else:
        p.temps_execution += 1
        if meilleur_priorite(liste_files) <= p.priorite:
            p.priorite += 1
            liste_files[p.priorite].append(p)
            return prioritaire(liste_files)
        else:
            p.priorite += 1
            return p
```

Exercice 3 — 8 points

| *Questions* | *Contenu et notions* | *Capacités exigibles / Niveau* | *Éléments de réponses et commentaires* |
|---|---|---|---|
| 1 | Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : `SELECT`, `FROM`, `WHERE`, `JOIN`. | N1 | `SELECT nom_patient, prenom`<br>`FROM Patient`<br>`WHERE age > 60` |
| 2 | Construire des requêtes d'insertion et de mise à jour à l'aide de : `UPDATE`, `INSERT`, `DELETE`. | N2 | `UPDATE Symptome`<br>`SET toux = 'Non'`<br>`WHERE nom_patient = 'Heartman';` |
| 3 | Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : `SELECT`, `FROM`, `WHERE`, `JOIN`. | N2 | `SELECT COUNT(*)`<br>`FROM Symptome JOIN Diagnostic`<br>`  ON Symptome.nom_patient = Diagnostic.nom_patient`<br>`WHERE toux = 'Oui' AND nom_maladie = 'Covid-19';` |
| 4 | Identifier les concepts définissant le modèle relationnel. (clé primaire) | N2 | L'employé souhaite ajouter dans la base de données un patient nommé Patrick Douglas Cette requête ne fonctionne pas car la clé primaire doit être unique, or un patient nommé Douglas est déjà présent. |
| 5 | Identifier les concepts définissant le modèle relationnel (clé primaire). | N1 | On peut utiliser le numéro de sécurité sociale comme clé primaire.<br>Alternatives : ajouter un identifiant unique par patient ou utiliser le couple (nom + prénom) comme clé primaire (même si il reste des possibilités d'homonymies) |
| 6 | Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits. ; Rechercher une clé dans un arbre de recherche, insérer une clé. | N1 | Le diagnostic est positif (+). |

| Exercice 3 | 8 points | | |
|---|---|---|---|
| 7 | Mise au point des programmes. Gestion des bugs. | N1 | On vérifie que le nœud n'est pas une feuille. |
| 8 | Vocabulaire de la programmation objet : classes, attributs, méthodes, objets. | N1 | Attributs : valeur, gauche, droit. Méthodes : est_feuille, symptome, diagnostic, **init** |
| 9 | Accéder aux attributs et méthodes d'une classe. ; Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits. ; Algorithmes sur les arbres binaires. | N2 | `def applique(arbre, patient):`<br>`  if arbre.est_feuille():`<br>`  return arbre.diagnostic()`<br>`  else:`<br>`  if patient[arbre.symptome()]:`<br>`  return applique(arbre.droit, patient)`<br>`  else:`<br>`  return applique(arbre.gauche, patient)` |
| 10 | Calculer la taille et la hauteur d'un arbre. | N1 | La taille est 31. |
| 11 | Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits. ; Algorithmes sur les arbres binaires. Analyser le fonctionnement d'un programme récursif. | N2 | L'arbre final possède 8 nœuds internes et 9 feuilles. L'arbre est donné dans la figure 1. On attentif au fait que l'arbre se termine sur un "+" en cas de fièvre. |

| Exercice 3 | 8 points | | |
|---|---|---|---|
| 12 | Accéder aux attributs et méthodes d'une classe. Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits. ; Algorithmes sur les arbres binaires ; Écrire un programme récursif. | N3 | `def reduire(self):`<br>`  """fonction récursive qui réduit la taille d'un arbre de décision sans changer les décisions prises"""`<br>`  if self.est_feuille():`<br>`  return`<br>`  self.gauche.reduire()`<br>`  self.droit.reduire()`<br>`  if (self.gauche.est_feuille()`<br>`  and self.droit.est_feuille()`<br>`  and self.gauche.diagnostic() == self.droit.diagnostic()):`<br>`  self.valeur = self.gauche.diagnostic()`<br>`  self.gauche = None`<br>`  self.droit = None` |
| 13 | Programmation Python. ; Écriture d'un entier positif dans une base. (1ère) | N1 | `def verifie(num_secu):`<br>`  n = num_secu // 100`<br>`  k = num_secu % 100`<br>`  return (k + n) % 97 == 0` |
| 14 | Programmation Python. ; Écriture d'un entier positif dans une base. (1ère) | N1 | `def cle(n):`<br>`  return 97 - (n % 97)` |

Tableau 1.

| Diagnostic | |
|---|---|
| nom_patient | nom_maladie |
| Heartman | Covid-19 |
| Douglas | Gastroentérite |
| Woods | Covid-19 |

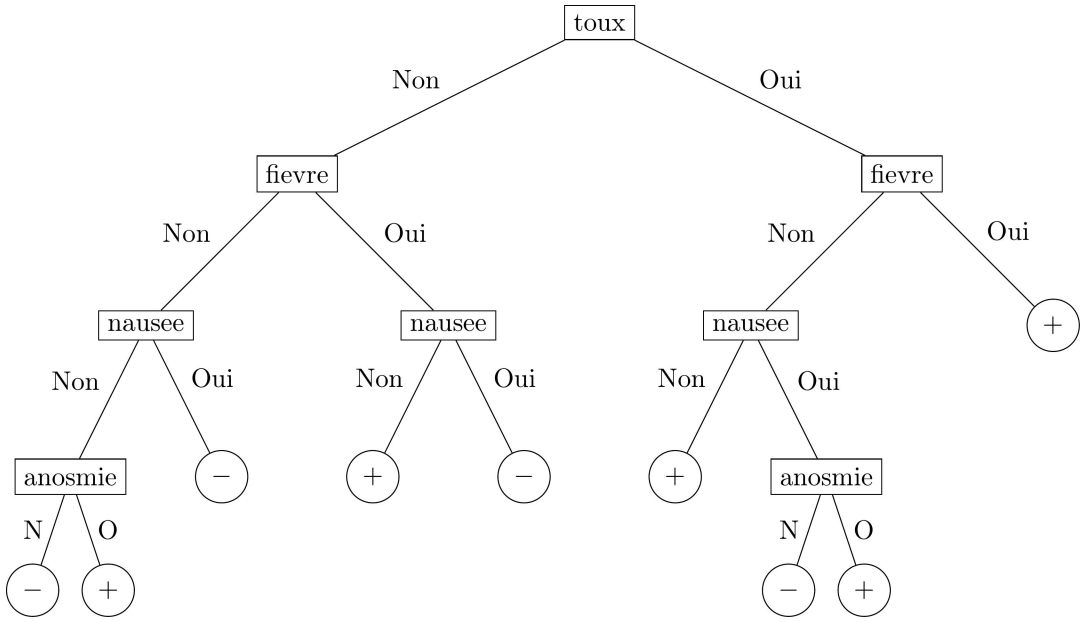


Figure 1: Arbre compressé