BACCALAURÉAT GÉNÉRAL

CORRECTION DE L'ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2024

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Durée de l'épreuve : 3 heures 30

Le sujet est composé de trois exercices indépendants.

Exercice 1 — 6 points

Questions	*Contenu et notions*	*Capacités exigibles / Niveau*	*Éléments de réponses et commentaires*
1	Connaître les états d'un processus	N1	prêt (en attente), élu (en exécution) et bloqué
2	Connaître les états d'un processus	N1	prêt et élu

3	Analyser et rectifier un programme Python	N1	Tout code équivalent à
			`def defile(self):`
			`    if self.contenu:`
			`        return self.contenu.pop(0)`

4	Appliquer un algorithme d'ordonnancement à la main	N2	`p1 p1 p3 p1 p2 p3 p4 p1 p2 p3 p4 p2 p3 p4 p3` 0,5 points pour `p1 p3 p2 p4 p1 p3 p2 p4 p1 p3 p2 p4 p1 p3 p3` (exécution d'un processus nouvellement crée avant de le mettre en file)

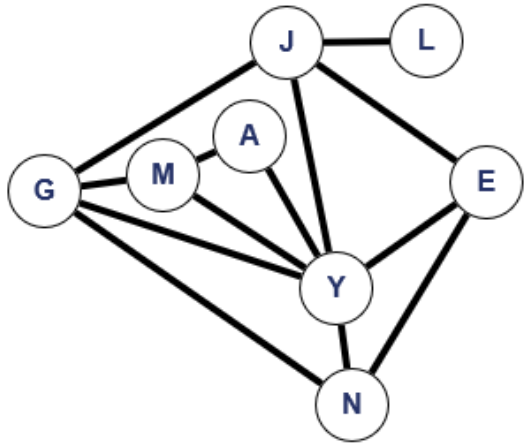
5	Compléter un code Python	N2	`class Ordonnanceur:`
			
			`    def __init__(self):`
			`     self.temps = 0`
			`     self.file = File()`
			
			`    def ajoute_nouveau_processus(self, proc):`
			`        self.file.enfile(proc)`
			
			`    def tourniquet(self):`
			`        self.temps += 1`
			`        if not self.file.est_vide():`
			`            proc = self.file.defile()`
			`            proc.execute_un_cycle()`
			`            if not proc.est_fini():`
			`                self.file.enfile(proc)`
			`            return proc.nom`
			`        else:`
			`            return None`

Exercice 1	6 points		

6	Résoudre un problème en autonomie et le programmer en Python	N3	Toute trace de recherche sera valorisée.
			`ordonnanceur = Ordonnanceur()`
			`fini = False`
			`while not fini:`
			`  if ordonnanceur.temps in depart_proc:`
			`  ordonnanceur.ajoute_nouveau_processus(depart_proc[ordonnanceur.temps`
			`  elu = ordonnanceur.tourniquet()`
			`  if elu:`
			`  print(elu, end=" ")`
			`  else:`
			`  fini = True`

7	Déceler un interblocage	N2	Lors du cycle 5, on observe le circuit `D -> Clavier -> B -> Fichier -> D` dans le graphe d'allocation.

Exercice 2 — 6 points

Questions	*Contenu et notions*	*Capacités exigibles / Niveau*	*Éléments de réponses et commentaires*
			
1	Structure de données; graphe	N1	

2	Structure de données; graphe	N1	`# sommets :  G, J, Y, E, N, M, A, L`
			`matrice_adj = [[0, 1, 1, 0, 1, 1, 0, 0], # G`
			`[1, 0, 1, 1, 0, 0, 0, 1], # J`
			`[1, 1, 0, 1, 1, 1, 1, 0], # Y`
			`[0, 1, 1, 0, 1, 0, 0, 0], # E`
			`[1, 0, 1, 1, 0, 0, 0, 0], # N`
			`[1, 0, 1, 0, 0, 0, 1, 0], # M`
			`[0, 0, 1, 0, 0, 1, 0, 0], # A`
			`[0, 1, 0, 0, 0, 0, 0, 0]] # L`

3	programmation Python	N1	`>>> position(sommets, 'G')`
			`0`
			`>>> position(sommets, 'Z')`
			`None`

Exercice 2	6 points		

4	programmation Python	N2	`1 def nb_amis(L, m, s):`
			`2  pos_s = position(L,s)`
			`3  if pos_s == None:`
			`4  return None`
			`5  amis = 0`
			`6  for i in range(len(m)):`
			`7  amis += m[pos_s][i]`
			`8  return amis`

5	programmation Python	N1	`>>> nb_amis(sommets, matrice_adj, 'G')` `4`
6	Représentation des données; dictionnaires	N1	`c` : clé et `v` : valeur

7	programmation Python ; POO	N3	`graphe = {'G' : ['J', 'Y', 'N', 'M'],`
			`'J' : ['G', 'Y', 'E', 'L'],`
			`'Y' : ['G', 'J', 'E', 'N', 'M', 'A'],`
			`'E' : ['J', 'Y', 'N'],`
			`'N' : ['G', 'Y', 'E'],`
			`'M' : ['G', 'Y', 'A'],`
			`'A' : ['Y', 'M'],`
			`'L' : ['J']`
			`}`

8	programmation Python	N1	`def nb_amis_dico(d, s):` `return len(d[s])`
9	algorithme sur les graphes; DFS	N3	`['L', 'J', 'G', 'Y', 'E', 'N']` ou `['J', 'G', 'Y', 'E', 'N']` ou Jade, Gabriel, Yanis, Emma, Nina dans le bon odre

10	programmation Python	N3	`1 def parcours_en_profondeur(d, s, visites = []):`
			`2  visites.append(s)`
			`3  for v in d[s]:`
			`4  if v not in visites:`
			`5  parcours_en_profondeur(d, v)`
			`6  return visites`

Exercice 3 — 8 points

Questions	*Contenu et notions*	*Capacités exigibles / Niveau*	*Éléments de réponses et commentaires*
1	traitement de données en table	N1	le point virgule
2	traitement de données en table	N1	certaines réponses ont des virgules

3			`def charger(nom_fichier):`
			`  with open(nom_fichier,'r') as fichier :`
			`  donnees = = list(csv.DictReader(fichier,delimiter=';'))`
			`  return donnees`

4	modularité	N1	`sleep`
5	programmation Python	N2	type `dict`

6			`flashcard = charger('flashcards.csv')`
			`d = choix_discipline(flashcard)`
			`c = choix_chapitre(flashcard, d)`
			`entrainement(flashcard, d, c)`

7	langage SQL	N1	`INSERT INTO boite` `VALUES (5, 'tous les 15 jours', 15)`

8	langage SQL	N1	`UPDATE flashcard`
			`SET reponse = '7 décembre 1941'`
			`WHERE id = 5`

9	langage SQL	N1	`SELECT lib` `FROM discipline`

10	langage SQL	N2	`SELECT chapitre.lib`
			`FROM chapitre`
			`JOIN discipline ON discipline.id = chapitre.id_ch`
			`WHERE discipline.lib = 'histoire'`

Exercice 3	8 points		

11	langage SQL	N3	`SELECT discipline.id`
			`FROM flashcard`
			`JOIN chapitre ON flashcard.id_ch = chapitre.id_ch`
			`JOIN discipline ON discipline.id = chapitre.id_ch`
			`WHERE discipline.lib = 'histoire'`

12	langage SQL	N2	`DELETE FROM flashcard` `WHERE flashcard.id_boite = 3`