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	application d'un algo donné	N1	`[10] -> [1, 9] -> [2, 8] -> [1, 2, 7] -> [1, 3, 6] -> [2, 3, 5] -> [1, 2, 3, 4]` qui est stable.
2	application d'un algo donné	N1	`[9] -> [1, 8] -> [2, 7] -> [1, 2, 6] -> [1, 3, 5] -> [2, 3, 4] -> [1, 2, 3, 3] -> [1, 2, 2, 4] -> [1, 1, 3, 4] -> [2, 3, 4] -> [1, 2, 3, 3]` et on tombe sur un cycle.
3	complétion d'un programme donné en suivant quelques indications	N2	ligne 15 : self.precedents.append(self.courant) ligne 22 : self.courant = suiv

4	programmation Python	N2	`if len(self.precedents) == 0:`
			`  return False`
			`else:`
			`  return self.precedents[len(self.precedents)-1] == self.courant`

5	programmation Python, un algo étant donné	N3	`def partie_stable(nb_cartes):`
			`  partie = Partie(nb_cartes)`
			`  while not partie.est_stable() and not partie.est_cyclique():`
			`  partie.suivante()`
			`  return partie.est_stable()`

6	choix des tests	N1	on peut proposer au minimum les quatre tests donnés en exemple :
			`assert partie_stable(15)`
			`assert partie_stable(10)`
			`assert not partie_stable(12)`
			`assert not partie_stable(9)`

7	réflexion sur les tris	N2	le tri par insertion est efficace lorsque la liste est déjà triée.

Exercice 2	6 points		
Questions	*Contenu et notions*	*Capacités exigibles*	*Éléments de réponses et commentaires*
1	Réseaux, adressage IP (Première)	N1	• 192.168.129.3 D • 192.168.194.2 AUCUN • 192.168.192.7 B • 192.168.72.25 AUCUN
2	Réseaux, adressage IP (Première)	N1	A : 2046, D : 510
3	Réseaux, adressage IP (Première)	N2	R1 : 192.168.64.1 (A), R3 : 192.168.192.2 (B)
4	Réseaux routage tables statiques	N1	cf Table 4
5	Réseaux routage protocole RIP	N2	cf Table 5
6	Réseaux, utilisation table de routage	N1	cf Table 6
7	Réseaux, découpage en sous réseaux (première ?)	N2	192.168.128.0/24 et 192.168.129.0/24
8	Réseaux routage protocole RIP	N3	cf Tables 8a et 8b

Table 4 : Table de routage de R2

Réseau	Masque	Passerelle	Hops
192.168.64.0	255.255.248.0	0.0.0.0	0
192.168.192.0	255.255.255.0	0.0.0.0	0
192.168.193.0	255.255.255.0	192.168.192.2 (R3)	1
192.168.128.0	255.255.254.0	192.168.64.1 (R1)	1

Table 5 : Table de routage de R4

Réseau	Masque	Passerelle	Hops
192.168.193.0	255.255.255.0	0.0.0.0	0
192.168.128.0	255.255.254.0	0.0.0.0	0
192.168.64.0	255.255.248.0	192.168.128.1 (R1)	1
192.168.192.0	255.255.255.0	192.168.193.1 (R3)	1

Table 6 : Routeurs traversés

Nœud à joindre	Routeurs traversés
192.168.193.17	R2, R3
192.168.129.17	R2, R1
192.168.65.17	R2
192.168.192.27	Aucun

Table 8a : Table de routage de R1

Réseau	Masque	Passerelle	Hops
192.168.64.0	255.255.248.0	0.0.0.0	0
192.168.192.0	255.255.255.0	192.168.64.2 (R2)	1
192.168.193.0	255.255.255.0	192.168.128.2 (R4)	1
192.168.128.0	**255.255.255.0**	0.0.0.0	0
192.168.129.0	**255.255.255.0**	**0.0.0.0**	**0**

Table 8b : Table de routage de R4

Réseau	Masque	Passerelle	Hops
192.168.193.0	255.255.255.0	0.0.0.0	0
192.168.128.0	**255.255.255.0**	0.0.0.0	0
192.168.64.0	255.255.248.0	192.168.128.1 (R1)	1
192.168.192.0	255.255.255.0	192.168.193.1 (R3)	1
192.168.129.0	**255.255.255.0**	**192.168.128.1 (R1)**	**0**

Exercice 3 — 6 points

Questions	*Contenu et notions*	*Capacités exigibles*	*Éléments de réponses et commentaires*
1	Prog Python. Accéder aux éléments d'une liste	N1	`print(tab_trace[-1][1])`

2	Prog Python. Création d'une liste de dictionnaires	N1	`def creation_trace(tab_trace):`
			`  trace = []`
			`  for enregistrement in tab_trace:`
			`  pt = {}`
			`  pt['lat'] = float(enregistrement[0])`
			`  pt['lon'] = float(enregistrement[1])`
			`  pt['alt'] = int(enregistrement[2])`
			`  pt['tsp'] = int(enregistrement[3])`
			`  trace.append(pt)`
			`  return trace`

3	Prog Python. Extraction d'informations d'une liste.	N1	`def valeurs_altitude(trace):`
			`  altitudes = [p['alt'] for p in trace]`
			`  return altitudes`
			ou
			`def valeurs_altitude(trace):`
			`  altitudes = []`
			`  for pt in trace:`
			`  altitudes.append(pt['alt'])`
			`  return altitudes`

4	Prog Python. Calcul arithmétique sur les éléments d'une liste	N2	`def denivele(altitudes):`
			`  cumule_positif = 0`
			`  cumule_negatif = 0`
			`  for indice in range(1, len(altitudes)):`
			`  denivele = altitudes[indice] - altitudes[indice - 1]`
			`  if denivele >= 0:`
			`  cumule_positif = cumule_positif + denivele`
			`  else:`
			`  cumule_negatif = cumule_negatif - denivele`
			`  return (cumule_positif, cumule_negatif)`

Exercice 3	8 points		

5	Prog Python. Calcul arithmétique sur les éléments du'ne liste	N2	`def asc_desc(trace):`
			`  temps_ascension = 0`
			`  temps_descente = 0`
			`  temps_plat = 0`
			`  for indice in range(1, len(trace)):`
			`  temps = trace[indice]['tsp'] - trace[indice - 1]['tsp']`
			`  denivele = trace[indice]['alt'] - trace[indice - 1]['alt']`
			`  if denivele > 0:`
			`  temps_ascension = temps_ascension + temps`
			`  elif denivele < 0:`
			`  temps_descente = temps_descente + temps`
			`  else:`
			`  temps_plat = temps_plat + temps`
			`  return (temps_ascension, temps_descente, temps_plat)`

6	Prog Python. Traduction formule trigo en code	N3	`def distance(p1, p2):`
			`  lat_A = math.radians(p1['lat'])`
			`  lon_A = math.radians(p1['lon'])`
			`  lat_B = math.radians(p2['lat'])`
			`  lon_B = math.radians(p2['lon'])`
			`  lecos = math.sin(lat_A) * math.sin(lat_B) + \`
			`  math.cos(lat_A) * math.cos(lat_B) * math.cos(lon_A - lon_B)`
			`  # Pb d'arrondi avec les données réelles, mais le sujet ne demande pas`
			`  # de penser à faire ça...`
			`  # lecos = max(-1, min(1, lecos))`
			`  distance_surface = 6371000 * math.acos(lecos)`
			`  return distance_surface`

7	Algo. Amélioration d'une méthode existante	N3	`def distance_precise(p1, p2):`
			`  distance_surface = distance(p1, p2)`
			`  distance_reelle = math.sqrt((p1['alt'] - p2['alt']) ** 2 + distance_surfac`
			`  return distance_reelle`

8	bdd	N1	`Mont Blanc | 21.3` `Mont Blanc | 171`
9	bdd	N2	`SELECT id, lieu FROM trails WHERE dist > 50;`
10	bdd	N2	`SELECT lieu, dist FROM trails WHERE denivele_p <> denivele_n;`
11	bdd	N2	`SELECT SUM(dist) from trails WHERE denivele_p > 1000;`