BACCALAURÉAT GÉNÉRAL

ÉPREUVE D’ENSEIGNEMENT DE SPÉCIALITÉ

**SESSION 2023**

**NUMÉRIQUE ET SCIENCES INFORMATIQUES**

**ÉPREUVE DU LUNDI 20 MARS 2023**

Durée de l’épreuve : **3 heures 30**

*L’usage de la calculatrice n’est pas autorisé.*

Dès que ce sujet vous est remis, assurez-vous qu’il est complet.

Ce sujet comporte 10 pages numérotées de 1/10 à 10/10.

**Le candidat traite les 3 exercices proposés**

**EXERCICE 1 (3 points)**

*Cet exercice porte sur la notion de base de données relationnelle et le langage SQL.*

On pourra utiliser les mots-clés SQL suivants : AND, FROM, INSERT, INTO, JOIN, ON, SELECT, SET, UPDATE, VALUES, WHERE.

Un grand magasin de meubles propose à ses clients un large choix de meubles. Les informations correspondantes sont rangées dans une base de données composée de trois relations.

Voici le schéma de deux de ces relations :

• Clients (id, nom, prenom, adresse, ville)

• Commandes (id, #idClient, #idMeuble, quantite, date)

Dans ce schéma :

- la clé primaire de chaque relation est définie par les attributs soulignés ;

- les attributs précédés de # sont les clés étrangères.

La troisième relation est appelée Meubles et concerne les meubles du magasin.

Le tableau de la figure 1 ci-dessous en présente un extrait :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | intitule | prix | stock | description |
| 62  63  74  98 | 'skap'  'skap'  'stol'  'hylla' | 69.99  69.99  39.99  99.99 | 2  3  10  0 | 'Armoire blanche 3 portes'  'Armoire noire 3 portes'  'Chaise en bois avec tissu bleu'  'Bibliothèque 5 étages blanche' |

Figure 1 – Extrait de la relation Meubles

**1.**Dans cette question, on s'intéresse au modèle relationnel.

**a.** Donner la caractéristique qu’un attribut doit avoir pour être choisi comme clé primaire.

**b.** Expliquer le rôle des deux clés étrangères de la relation Commandes.

**c.** Donner le schéma relationnel de la relation Meubles en précisant la clé primaire et les éventuelles clés étrangères.

**2.** En vous basant uniquement sur les données du tableau de la figure 1, donner le résultat de la requête suivante :

|  |
| --- |
| SELECT id, stock, description  FROM Meubles  WHERE intitule = 'skap'; |

**3.** Donner la requête SQL permettant d’afficher les noms et prénoms des clients habitant à Paris.

**4.** Le magasin vient de recevoir des meubles dont l’intitulé est 'hylla' et dont l’attribut id dans la relation Meubles vaut 98. Le stock de ces meubles est alors de 50.

Recopier et compléter la requête SQL ci-dessous qui permet de mettre à jour la base de données.

|  |
| --- |
| UPDATE ...  SET ...  WHERE ... |

**5.**Le magasin vient d’ajouter à son catalogue un nouveau meuble dont les caractéristiques sont les suivantes :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | intitule | prix | stock | description |
| 65 | 'matta' | 95.99 | 25 | 'Tapis vert à pois rouges' |

Donner la requête SQL qui permet d’ajouter cet article à la relation Meubles.

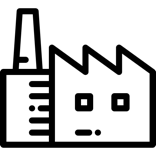
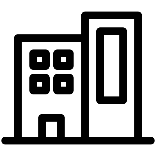
**6.**Donner la requête SQL permettant de récupérer le nom et le prénom des différents clients qui ont passé une commande le 30 avril 2021.

On précise que, dans la relation Commandes, les dates sont des chaînes de caractères, par exemple '21/08/2002'.

**EXERCICE 2 (3 points)**

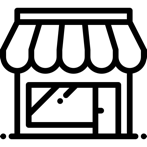
*Cet exercice porte sur les réseaux.*

Le réseau d’une entreprise dispose de quatre sites (SiteA, SiteB, SiteC et SiteD) et de cinq routeurs (R1, R2, R3, R4 et R5). La figure 1 en donne une représentation.



**SiteA**

**SiteB**

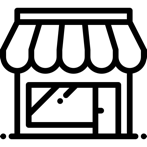


**Liaison1**

**Liaison3**

**Liaison4**

**Liaison2**



**SiteC**

**SiteD**

**Liaison5**

**Liaison6**

Figure 1 – Représentation du réseau de l’entreprise

**1.** Justifier brièvement que ce réseau sera encore fonctionnel même si l’une des liaisons (Liaison1, Liaison2, …, Liaison6) est coupée.

**2.** Dans un premier temps, les tables de routage des routeurs sont configurées manuellement (voir figure 2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Routeur R1** | | | **Destination** | **Suivant** | | SiteA | Local | | SiteB | R2 | | SiteC | R3 | | SiteD | R3 | | |  |  | | --- | --- | | **Routeur R2** | | | **Destination** | **Suivant** | | SiteA | R1 | | SiteB | Local | | SiteC | R3 | | SiteD | R3 | | |  |  | | --- | --- | | **Routeur R3** | | | **Destination** | **Suivant** | | SiteA | R1 | | SiteB | R2 | | SiteC | R4 | | SiteD | R4 | |
| |  |  | | --- | --- | | **Routeur R4** | | | **Destination** | **Suivant** | | SiteA | R3 | | SiteB | R3 | | SiteC | R5 | | SiteD | Local | | |  |  | | --- | --- | | **Routeur R5** | | | **Destination** | **Suivant** | | SiteA | R4 | | SiteB | R4 | | SiteC | Local | | SiteD | R4 | |  |

Figure 2 – Tables de routage des routeurs R1, R2, R3, R4 et R5

Indiquer le chemin suivi par les paquets lorsqu’une information est envoyée de SiteB à SiteC.

**3.** Afin d’optimiser la maintenance du réseau, les tables de routage sont configurées automatiquement en utilisant le protocole RIP.

Pour le protocole RIP, le chemin est construit de façon à minimiser le nombre de routeurs traversés.

Recopier et compléter la table de routage RIP du routeur R1 (voir figure 3).

|  |  |  |
| --- | --- | --- |
| **Routeur R1 (RIP)** | | |
| **Destination** | **Suivant** | **Nombre de sauts** |
| SiteA | Local | 0 |
| SiteB |  |  |
| SiteC |  |  |
| SiteD |  |  |

Figure 3 – Table de routage RIP du routeur R1

**4.** La liaison Liaison2 a un débit très inférieur aux autres liaisons. Expliquer pourquoi le choix du protocole RIP n’est pas judicieux.

**5.** On considère maintenant que les tables de routage sont configurées en utilisant le protocole OSPF.

Pour le protocole OSPF, le chemin est construit de façon à minimiser le coût. Le coût d’un chemin est la somme des coûts des liaisons à parcourir. Pour une liaison, la relation entre le coût (sans unité) et le débit *D* (en bit/s) est donnée par :

Par convention, le coût d’une liaison directe entre un routeur et un site est 0.

On donne le coût des liaisons dans la figure 4.

|  |  |
| --- | --- |
| **Liaison** | **Coût** |
| Liaison1 | 100 000 |
| Liaison2 | 1 000 000 |
| Liaison3 | 5 |
| Liaison4 | 50 000 |
| Liaison5 | 5 |
| Liaison6 | 10 |

Figure 4 – Table des coûts des liaisons

**a.** Indiquer la liaison dont le débit est le plus faible.

**b.** Donner la liste des quatre chemins possibles pour aller de SiteA à SiteC sans utiliser deux fois le même routeur et calculer le coût de chacun de ces chemins.

**c.** Recopier et compléter la table de routage OSPF du routeur R1 (voir figure 5).

|  |  |  |
| --- | --- | --- |
| **Routeur R1 (OSPF)** | | |
| **Destination** | **Suivant** | **Coût total du chemin** |
| SiteA | Local | 0 |
| SiteB |  |  |
| SiteC |  |  |
| SiteD |  |  |

Figure 5 – Table de routage OSPF du routeur R1

**EXERCICE 3 (6 points)**

*Cet exercice traite de programmation orientée objet en Python et d’algorithmique.*

Un pays est composé de différentes régions. Deux régions sont voisines si elles ont au moins une frontière en commun. L'objectif est d'attribuer une couleur à chaque région sur la carte du pays sans que deux régions voisines aient la même couleur et en utilisant le moins de couleurs possibles.

La figure 1 ci-dessous donne un exemple de résultat de coloration des régions de la France métropolitaine.

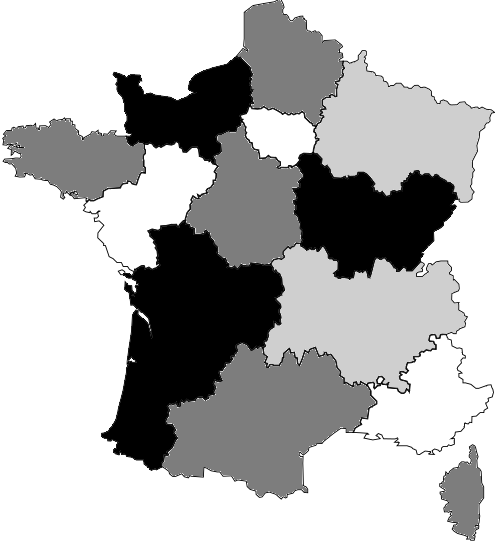


Figure 1 – Carte coloriée des régions de France métropolitaine

On rappelle quelques fonctions et méthodes des tableaux (le type list en Python) qui pourront être utilisées dans cet exercice :

• len(tab) : renvoie le nombre d'éléments du tableau tab ;

• tab.append(elt) : ajoute l'élément elt en fin de tableau tab ;

• tab.remove(elt) : enlève la première occurrence de elt de tab si elt est dans tab. Provoque une erreur sinon.

Exemple :

• len([1, 3, 12, 24, 3]) renvoie 5 ;

• avec tab = [1, 3, 12, 24, 3], l’instruction tab.append(7) modifie tab en [1, 3, 12, 24, 3, 7] ;

• avec tab = [1, 3, 12, 24, 3], l’instruction tab.remove(3) modifie tab en [1, 12, 24, 3].

Les deux parties de cet exercice forment un ensemble. Cependant, il n’est pas nécessaire d’avoir répondu à une question pour aborder la suivante. En particulier, on pourra utiliser les méthodes des questions précédentes même quand elles n’ont pas été codées.

Pour chaque question, toute trace de réflexion sera prise en compte.

**Partie 1**

On considère la classe Region qui modélise une région sur une carte et dont le début de l'implémentation est :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | class Region:  '''Modélise une région d'un pays sur une carte.'''  def \_\_init\_\_(self, nom\_region):  '''  initialise une région   : param nom\_region (str) le nom de la région  '''  self.nom = nom\_region   # tableau des régions voisines, vide au départ  self.tab\_voisines = []   # tableau des couleurs disponibles pour colorier la région   self.tab\_couleurs\_disponibles = ['rouge', 'vert', 'bleu', 'jaune', 'orange', 'marron']   # couleur attribuée à la région et non encore choisie au départ  self.couleur\_attribuee = None |

**1.** Associer, en vous appuyant sur l’extrait de code précédent, les noms nom, tab\_voisines, tab\_couleurs\_disponibles et couleur\_attribuee au terme qui leur correspond parmi : *objet*, *attribut*, *méthode* ou *classe*.

**2.** Indiquer le type du paramètre nom\_region de la méthode \_\_init\_\_ de la classe Region.

**3.** Donner une instruction permettant de créer une instance nommée ge de la classe Region correspondant à la région dont le nom est « Grand Est ».

**4.** Recopier et compléter la ligne 6 de la méthode de la classe Region ci-dessous :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | def renvoie\_premiere\_couleur\_disponible(self):  '''  Renvoie la première couleur du tableau des couleurs disponibles supposé non vide.  : return (str)  '''  return ... |

**5.** Recopier et compléter la ligne 6 de la méthode de la classe Region ci-dessous :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | def renvoie\_nb\_voisines(self) :  '''  Renvoie le nombre de régions voisines.  : return (int)   '''  return ... |

**6.** Compléter la méthode de la classe Region ci-dessous à partir de la ligne 6 :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | def est\_coloriee(self):  '''  Renvoie True si une couleur a été attribuée à cette région et False sinon.  : return (bool)  '''  ... |

**7.** Compléter la méthode de la classe Region ci-dessous à partir de la ligne 8 :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | def retire\_couleur(self, couleur):  '''  Retire couleur du tableau de couleurs disponibles de la région si elle est dans ce tableau. Ne fait rien sinon.  : param couleur (str)  : ne renvoie rien  : effet de bord sur le tableau des couleurs disponibles  '''  ... |

**8.** Compléter la méthode de la classe Region ci-dessous, à partir de la ligne 7, en utilisant une boucle :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | def est\_voisine(self, region):  '''  Renvoie True si la region passée en paramètre est une voisine et False sinon.  : param region (Region)  : return (bool)  '''  ... |

**Partie 2**

Dans cette partie :

- on considère qu’on dispose d’un ensemble d’instances de la classe Region pour lesquelles l’attribut tab\_voisines a été renseigné ;

- on pourra utiliser les méthodes de la classe Region évoquées dans les questions de la partie 1 :

• renvoie\_premiere\_couleur\_disponible

• renvoie\_nb\_voisines

• est\_coloriee

• retire\_couleur

• est\_voisine

On a créé une classe Pays :

- cette classe modélise la carte d’un pays composé de régions ;

- l’unique attribut tab\_regions de cette classe est un tableau (type list en Python) dont les éléments sont des instances de la classe Region.

**9.** Recopier et compléter la méthode de la classe Pays ci-dessous à partir de la ligne 7 :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | def renvoie\_tab\_regions\_non\_coloriees(self):  '''  Renvoie un tableau dont les éléments sont les régions du pays sans couleur attribuée.  : return (list) tableau d’instances de la classe Region  '''  ... |

**10.** On considère la méthode de la classe Pays ci-dessous.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | def renvoie\_max(self):  nb\_voisines\_max = -1  region\_max = None  for reg in self.renvoie\_tab\_regions\_non\_coloriees():  if reg.renvoie\_nb\_voisines() > nb\_voisines\_max:  nb\_voisines\_max = reg.renvoie\_nb\_voisines()  region\_max = reg  return region\_max |

**a.** Expliquer dans quel cas cette méthode renvoie None.

**b.** Indiquer, dans le cas où cette méthode ne renvoie pas None, les deux particularités de la région renvoyée.

**11.** Coder la méthode colorie(self) de la classe Pays qui choisit une couleur pour chaque région du pays de la façon suivante :

• On récupère la région non coloriée qui possède le plus de voisines.

• Tant que cette région existe :

🞟 La couleur attribuée à cette région est la première couleur disponible dans son tableau de couleurs disponibles.

🞟 Pour chaque région voisine de la région :

🢭 si la couleur choisie est présente dans le tableau des couleurs disponibles de la région voisine alors on la retire de ce tableau.

🞟 On récupère à nouveau la région non coloriée qui possède le plus de voisines.