BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

BAC BLANC 2022

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice et du dictionnaire n'est pas autorisé.

Exercice 1 : *Cet exercice porte sur les réseaux et les protocoles de routage.*

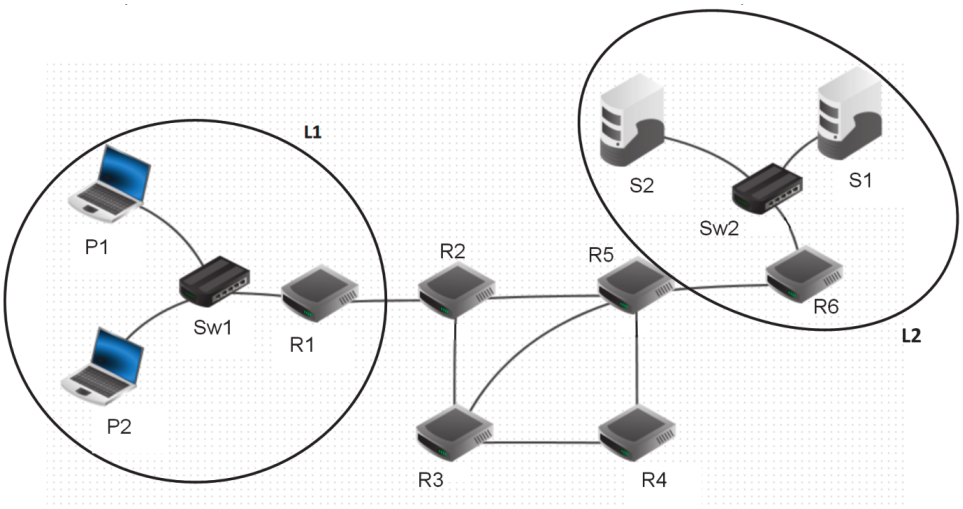

Figure 1 : Réseau d'entreprise

La figure 1 ci-dessus représente le schéma d'un réseau d'entreprise. Il y figure deux réseaux locaux L1 et L2. Ces deux réseaux locaux sont interconnectés par les routeurs R2, R3, R4 et R5. Le réseau local L1 est constitué des PC portables P1 et P2 connectés à la passerelle R1 par le switch Sw1. Les serveurs S1 et S2 sont connectés à la passerelle R6 par le switch Sw2. Le tableau 1 suivant indique les adresses IPv4 des machines constituants le réseau de l'entreprise.

Nom	Type	Adresse IPv4
R1	routeur (passerelle)	Interface 1 : 192.168.1.1/24 Interface 2 : 10.1.1.2/24
R2	routeur	Interface 1 : 10.1.1.1/24 Interface 2 : 10.1.2.1/24 Interface 3 : 10.1.3.1/24
R3	routeur	Interface 1 : 10.1.2.2/24 Interface 2 : 10.1.4.2/24 Interface 3 : 10.1.5.2/24
R4	routeur	Interface 1 : 10.1.5.1/24 Interface 2 : 10.1.6.1/24
R5	routeur	Interface 1 : 10.1.3.2/24 Interface 2 : 10.1.4.1/24 Interface 3 : 10.1.6.2/24 Interface 4 : 10.1.7.1/24
R6	routeur (passerelle)	Interface 1 : 172.16.0.1/16 Interface 2 : 10.1.7.2/24
P1	ordinateur portable	192.168.1.40/24
P2	ordinateur portable	192.168.1.46/24
S1	serveur	172.16.8.10/16
S2	serveur	172.16.9.12/16

Tableau 1 : adresses IPv4 des machines

Rappels et notations

Rappelons qu'une adresse IP est composée de 4 octets, soit 32 bits. Elle est notée X1.X2.X3.X4, où X1, X2, X3 et X4 sont les valeurs des 4 octets. Dans le tableau 1, les valeurs des 4 octets ont été converties en notation décimale.

La notation X1.X2.X3.X4/n signifie que les n premiers bits de poids forts de l'adresse IP représentent la partie « réseau », les bits suivants de poids faibles représentent la partie « machine ».

Toutes les adresses des machines connectées à un réseau local ont la même partie réseau. L'adresse IP dont tous les bits de la partie « machine » sont à 0 est appelée « adresse du réseau ». L'adresse IP dont tous les bits de la partie « machine » sont à 1 est appelée « adresse de diffusion ».

1) a) Quelles sont les adresses des réseaux locaux L1 et L2 ?

   b) Donner la plus petite et la plus grande adresse IP valides pouvant être attribuées à un ordinateur portable ou un serveur sur chacun des réseaux L1 et L2 sachant que l'adresse du réseau et l'adresse de diffusion ne peuvent pas être attribuées à une machine.

   c) Combien de machines peut-on connecter au maximum à chacun des réseaux locaux L1 et L2 ? On donne ci-dessous les valeurs de quelques puissances de 2 ?

2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
64	128	256	512	1 024	2 048	4 096	8 192	16 384	32 768	65 536	131 072

2) a) Expliquer l'utilité d'avoir plusieurs chemins possibles reliant les réseaux L1 et L2.

   b) Quel est le chemin le plus court en nombre de sauts pour relier R1 et R6 ? Donner le nombre de sauts de ce chemin et préciser les routeurs utilisés.

   c) La bande passante d'une liaison Ether (quantité d'information qui peut être transmise en bits/s) est de 10^7 bits/s et celle d'une liaison FastEther est de 10^8 bits/s. Le coût d'une liaison est défini par $10^8/d$, où d est sa bande passante en bits/s.

Laison	R1-R2	R2-R5	R5-R6	R2-R3	R3-R4	R4-R5	R3-R5
Type	Ether	Ether	Ether	FastEther	FastEther	FastEther	Ether

Tableau 2 : type des liaisons entre les routeurs

   Quel est le chemin reliant R1 et R6 qui a le plus petit coût ? Donner le coût de ce chemin et préciser les routeurs utilisés.

3) On donne ci-dessous les tables de routage des routeurs R1 et R2 au démarrage du réseau. Celles de R5 et R6 se trouvent en **annexe**. Compléter les lignes laissées vides des tables de routage des routeurs R5 et R6 pour que les échanges entre les ordinateurs des réseaux L1 et L2 se fassent en empruntant le chemin le plus court en nombre de sauts.

Table de routage de R1 :

IP Réseau de destination	Passerelle	Interface
192.168.1.0/24	–	Interface 1
10.1.1.0/24	–	Interface 2
0.0.0.0/0	10.1.1.1	Interface 2

Table de routage de R2 :

IP Réseau de destination	Passerelle	Interface
10.1.1.0/24	–	Interface 1
10.1.2.0/24	–	Interface 2
10.1.3.0/24	–	Interface 3
192.168.1.0/24	10.1.1.2	Interface 1
172.16.0.0/16	10.1.3.2	Interface 3

Exercice 2 : *Cet exercice porte sur les arbres binaires de recherche.*
Dans cet exercice, les arbres binaires de recherche ne peuvent pas comporter plusieurs fois la même clé. De plus, un arbre binaire de recherche limité à un nœud a une hauteur de 1.
On considère l'arbre binaire de recherche représenté ci-dessous (figure 2), où "val" représente un entier :

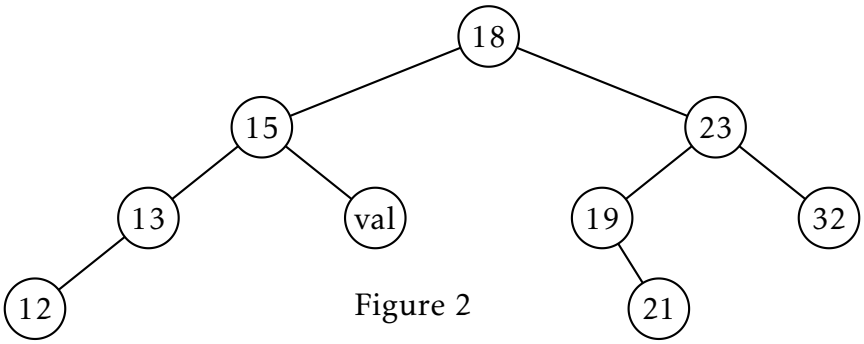

Figure 2

1) a) Donner le nombre de feuilles de cet arbre et préciser leur valeur (étiquette).
   b) Donner le sous-arbre gauche du nœud 23.
   c) Donner la hauteur et la taille de l'arbre de la figure 2.
   d) Donner les valeurs entières possibles de val pour cet arbre binaire de recherche.

On suppose, pour la suite de cet exercice, que "val" est égal à 16.

2) On rappelle qu'un parcours infixe depuis un nœud consiste, dans l'ordre, à faire un parcours infixe sur le sous-arbre gauche, afficher le nœud puis faire un parcours infixe sur le sous-arbre droit.
   Dans le cas d'un parcours suffixe, on fait un parcours suffixe sur le sous-arbre gauche puis un parcours suffixe sur le sous-arbre droit, avant d'afficher le nœud.
   a) Donner les valeurs d'affichage des nœuds dans le cas du parcours infixe de l'arbre.
   b) Donner les valeurs d'affichage des nœuds dans le cas du parcours suffixe de l'arbre.
3) On considère la classe `Noeud`, définie en Python, donnée en **annexe**.
   a) Représenter l'arbre construit suite à l'exécution de l'instruction suivante :

```
racine = Noeud(18)
racine.insere_tout([12, 13, 15, 16, 19, 21, 32, 23])
```

   b) Ecrire les deux instructions permettant de construire l'arbre de la figure 2. On rappelle que le nombre "val" est égal à 16.
   c) On considère l'arbre tel qu'il est présenté sur la figure 2. Déterminer l'ordre d'exécution des blocs (repérés de 1 à 3) suite à l'application de la méthode `insere(19)` au nœud racine de cet arbre.
4) Ecrire une méthode `recherche(self, v)` qui prend en argument un entier `v` et renvoie la valeur `True` si cet entier est une étiquette de l'arbre, `False` sinon.

Exercice 3 : *Thèmes abordés : bases de données*

Vous trouverez, en **annexe**, des rappels sur le langage SQL.

Un club de handball souhaite regrouper efficacement toutes ses informations. Il utilise pour cela des bases de données relationnelles afin d'avoir accès aux informations classiques sur les licenciés du club ainsi que sur les matchs du championnat. Le langage SQL a été retenu.

On suppose dans l'exercice que tous les joueurs d'une équipe jouent à chaque match de l'équipe.

La structure de la base de données est composée des deux tables (ou relations) suivantes:

Table `licencies`	
Attributs	**Types**
`id_licencie`	INT
`prenom`	VARCHAR
`nom`	VARCHAR
`annee_naissance`	INT
`equipe`	VARCHAR

Table `matchs`	
Attributs	**Types**
`id_matchs`	INT
`equipe`	VARCHAR
`adversaire`	VARCHAR
`lieu`	VARCHAR
`date`	DATE

Ci-dessous un exemple de ce que l'on peut trouver dans la base de données :

Exemple **non exhaustif** d'entrées de la table `licencies`

`id_licencie`	`prenom`	`nom`	`annee_naissance`	`equipe`
63	Jean-Pierre	Masclef	1965	Vétérans
102	Eva	Cujon	1992	Femmes 1
125	Emile	Alinio	2000	Hommes 2
247	Ulysse	Trentain	2008	-12 ans

Exemple **non exhaustif** d'entrées de la table `matchs`

`id_match`	`equipe`	`adversaire`	`lieu`	`date`
746	-16 ans	PHC	Domicile	2021-06-19
780	Vétérans	PHC	Exterieur	2021-06-26
936	Hommes 3	LSC	Exterieur	2021-06-20
1032	-19 ans	LOH	Exterieur	2021-05-22
1485	Femmes 2	CHM	Domicile	2021-05-02
1512	Vétérans	ATC	Domicile	2021-04-12

1) a) L'attribut `nom` de la table `licencies` pourrait-il servir de clé primaire ? **Justifier.**
   b) **Citer** un autre attribut de cette table qui pourrait servir de clé primaire.
2) a) **Expliquer** ce que renvoie la requête SQL suivante :

```sql
SELECT prenom, nom FROM licencies WHERE equipe = "-12 ans"
```

   b) **Que renvoie** la requête précédente si `prenom, nom` est remplacé par une étoile (*) ?
   c) **Écrire** la requête qui permet l'affichage des dates de tous les matchs joués à domicile de l'équipe *Vétérans*.
3) **Écrire** la requête qui permet d'inscrire dans la table `licencies`, *Jean Lavenu* né en *2001* de l'équipe *Hommes 2* et qui aura comme numéro de licence *287* dans ce club.
4) On souhaite mettre à jour les données de la table `licencies` du joueur *Joseph Cuviller*, déjà inscrit. Il était en équipe *Hommes 2* et il est maintenant en équipe *Vétérans*. Afin de modifier la table dans ce sens, proposer la requête adéquate.
5) Pour obtenir le nom de tous les licenciés qui jouent contre le LSC le 19 juin 2021, compléter la requête se trouvant en **annexe**.

Exercice 4 : *Principaux thèmes abordés : structure de données (programmation objet) et langages et programmation (spécification).*

Une entreprise fabrique des yaourts qui peuvent être soit nature (sans arôme), soit aromatisés (fraise, abricot ou vanille).

Pour pouvoir traiter informatiquement les spécificités de ce produit, on va donc créer une classe Yaourt qui possèdera un certain nombre d'attributs :

- Son genre : nature ou aromatisé
- Son arôme : fraise, abricot, vanille ou aucun
- Sa date de durabilité minimale (DDM) exprimée par un entier compris entre 1 et 365 (on ne gère pas les années bissextiles). Par exemple, si la DDM est égale à 15, la date de durabilité minimale est le 15 janvier.

On va créer également des méthodes permettant d'interagir avec l'objet Yaourt pour attribuer un arôme ou récupérer un genre par exemple. On peut représenter cette classe par le tableau de spécifications ci-dessous :

Yaourt	
ATTRIBUTS :	**MÉTHODES :**
• genre	• `Construire(arome, duree)`
• arome	• `Obtenir_Arome()`
• duree	• `Obtenir_Genre()`
	• `Obtenir_Duree()`
	• `Attribuer_Arome(arome)`
	• `Attribuer_Duree(duree)`
	• `Attribuer_Genre(arome)`

1) La classe `Yaourt` est déclarée en Python à l'aide du mot-clé **`class`** :

```
class Yaourt:
    """ Classe définissant un yaourt caractérisé par :
    - son arome
    - son genre
    - sa durée de durabilité minimale"""
```

Le code existant se trouve à la fin de l'exercice. Il indique l'endroit des codes à produire dans les questions suivantes.

a) Quelles sont les assertions à prévoir pour vérifier que l'arôme et la durée correspondent bien à des valeurs acceptables. Il faudra aussi expliciter les commentaires qui seront renvoyés.

Pour rappel :

- L'arôme doit prendre comme valeur `'fraise'`, `'abricot'`, `'vanille'` ou `'aucun'`.
- Sa date de durabilité minimale (DDM) est une valeur positive.

b) Pour créer un yaourt, on exécutera la commande suivante :

```
Mon_Yaourt = Yaourt('fraise',24)
```

Quelle valeur sera affectée à l'attribut genre associé à `Mon_Yaourt` ?

c) Écrire en python une fonction `GetArome(self)`, renvoyant l'arôme du yaourt créé.

2) On appelle mutateur une méthode permettant de modifier un ou plusieurs attributs d'un objet.

Sur votre copie, écrire en Python le mutateur `SetArome(self, arome)` permettant de modifier l'arôme du yaourt.

On veillera à garder une cohérence entre l'arôme et le genre.

3) On veut créer une pile contenant le stock de yaourts. Pour cela il faut tout d'abord créer une pile vide :

```
def creer_pile():
    pile = [ ]
    return pile
```

a) Créer une fonction `empiler(p, yaourt)` qui renvoie la pile p après avoir ajouté un objet de type `Yaourt` à la fin.
b) Créer une fonction `depiler(p)` qui renvoie l'objet à dépiler.
c) Créer une fonction `estVide(p)` qui renvoie **`True`** si la pile est vide et **`False`** sinon.
d) Qu'affiche le bloc de commandes suivantes ci-dessous ?

```
mon_Yaourt1 = Yaourt('aucun', 18)
mon_Yaourt2 = Yaourt('fraise', 24)
ma_pile = creer_pile()
empiler(ma_pile, mon_Yaourt1)
empiler(ma_pile, mon_Yaourt2)
print(depiler(ma_pile).GetDuree())
print(estVide(ma_pile))
```

```
class Yaourt:
    def __init__(self, arome, duree):
        # ** Assertions : question 1.a. à compléter sur votre copie
        self.__arome = arome
        self.__duree = duree
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'

    # ** Méthode GetArome(self) question 1.c. à compléter sur votre copie

    def GetDuree(self):
        return (self.__duree)

    def GetGenre(self):
        return (self.__genre)

    def SetDuree(self, duree):
        # **** Mutateur de durée
        if duree > 0 :
            self.__duree = duree

    # ** Mutateur d'arôme SetArome(self,arome) - question 2. à compléter sur votre copie

    def __SetGenre(self, arome):
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'
```

Exercice 5 : *Cet exercice porte sur la programmation en général et la récursivité en particulier.*
On considère un tableau de nombres de n lignes et p colonnes.
Les lignes sont numérotées de 0 à $n-1$ et les colonnes sont numérotées de 0 à $p-1$. La case en haut à gauche est repérée par (0,0) et la case en bas à droite par $(n-1,p-1)$.
On appelle chemin une succession de cases allant de la case (0,0) à la case $(n-1,p-1)$, en n'autorisant que des déplacements case par case : soit vers la droite, soit vers le bas.
On appelle somme d'un chemin la somme des entiers situés sur ce chemin.
Par exemple, pour le tableau T suivant :

4	**1**	**1**	3
2	0	**2**	1
3	1	**5**	**1**

- Un chemin est (0,0), (0,1), (0,2), (1,2), (2,2), (2,3) (en gras sur le tableau).
- La somme du chemin précédent est 14.
- (0,0), (0,2), (2,2), (2,3) n'est pas un chemin.

L'objectif de cet exercice est de déterminer la somme maximale pour tous les chemins possibles allant de la case (0,0) à la case $(n-1,p-1)$.

1) On considère tous les chemins allant de la case (0,0) à la case (2,3) du tableau T donné en exemple.
   a) Un tel chemin comprend nécessairement 3 déplacements vers la droite. Combien de déplacements vers le bas comprend-il ?
   b) La longueur d'un chemin est égal au nombre de cases de ce chemin. Justifier que tous les chemins allant de (0,0) à (2,3) ont une longueur égale à 6.
2) En listant tous les chemins possibles allant de (0,0) à (2,3) du tableau T, déterminer un chemin qui permet d'obtenir la somme maximale et la valeur de cette somme.
3) On veut créer le tableau T′ où chaque élément $T'[i][j]$ est la somme maximale pour tous les chemins possibles allant de (0,0) à (i,j).
   a) Compléter le tableau T′ se trouvant annexe associé au tableau T ci-dessous.

$T =$

4	1	1	3
2	0	2	1
3	1	5	1

$T' =$

4	5	6	?
6	?	8	10
9	10	?	16

   b) Justifier que si j est différent de 0, alors : $T'[0][j] = T[0][j] + T'[0][j-1]$
4) Justifier que si i et j sont différents de 0, alors :

$$T'[i][j] = T[i][j] + \max(T'[i-1][j], T'[i][j-1])$$

5) On veut créer la fonction récursive `somme_max` ayant pour paramètres un tableau T, un entier i et un entier j. Cette fonction renvoie la somme maximale pour tous les chemins possibles allant de la case (0,0) à la case (i,j).
   a) Quel est le cas de base, à savoir le cas qui est traité directement sans faire appel à la fonction `somme_max` ? Que renvoie-t-on dans ce cas ?
   b) À l'aide de la question précédente, écrire en Python la fonction récursive `somme_max`.
   c) Quel appel de fonction doit-on faire pour résoudre le problème initial ?

Annexe à rendre avec la copie

Nom et prénom :

Exercice 1 : Compléter les tables suivantes :

Table de routage de R5 :

IP Réseau de destination	Passerelle	Interface
10.1.3.0/24	–	Interface 1
10.1.4.0/24	–	Interface 2
10.1.6.0/24	–	Interface 3
10.1.7.0/24	–	Interface 4

Table de routage de R6 :

IP Réseau de destination	Passerelle	Interface
172.16.0.0/16	–	Interface 1

Exercice 2 :

```
class Noeud():
    def __init__(self, v):
        self.ag = None
        self.ad = None
        self.v = v

    def insere(self, v):
        n = self
        est_insere = False
        while not est_insere :
            if v == n.v:
                est_insere = True       ## Bloc 1
            elif v < n.v:
                if n.ag != None:        ##
                    n = n.ag             #
                else:                    # Bloc 2
                    n.ag = Noeud(v)      #
                    est_insere = True   ##
            else:
                if n.ad != None:        ##
                    n = n.ad             #
                else:                    # Bloc 3
                    n.ad = Noeud(v)      #
                    est_insere = True   ##

    def insere_tout(self, vals):
        for v in vals:
            self.insere(v)
```

Exercice 3 : Rappels sur SQL

- **Types de données**

CHAR(t) VARCHAR(t) TEXT	Texte fixe de t caractères. Texte de t caractères variables. Texte de 65 535 caractères max.
INT	Nombre entier de -2^{31} à $2^{31}-1$ (signé) ou de 0 à $2^{32}-1$ (non signé)
FLOAT	Réel à virgule flottante
DATE DATETIME	Date format AAAA-MM-JJ Date et heure format AAAA-MM-JJHH:MI:SS

- **Quelques exemples de syntaxe SQL :**
  - Insérer des enregistrements :

```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1, valeur2)
```

Exemple :

```
INSERT INTO client (id_client, nom, prenom) VALUES (112,'Dehnou', 'Danièle')
```

  - Modifier des enregistrements :

```
UPDATE Table SET attribut1=valeur1, attribut2=valeur2 WHERE Selecteur
```

  - Supprimer des enregistrements :

```
DELETE FROM Table WHERE Selecteur
```

  - Sélectionner des enregistrements :

```
SELECT attributs FROM Table WHERE Selecteur
```

  - Effectuer une jointure :

```
SELECT attributs FROM TableA
JOIN TableB ON TableA.cle1=TableB.cle2
WHERE Selecteur
```

5) Requête à remplir :

```
SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE ..........................
```

Exercice 5 :

$T =$

4	1	1	3
2	0	2	1
3	1	5	1

$T' =$

4	5	6	
6		8	10
9	10		16