BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2023

NUMÉRIQUE ET SCIENCES INFORMATIQUES

JOUR 2

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 9 pages numérotées de 1/9 à 9/9.

Le candidat traite les 3 exercices proposés.

EXERCICE 1 (4 points)

Cet exercice porte sur l'adressage IP et les protocoles de routage.

Partie A : L'adressage IP

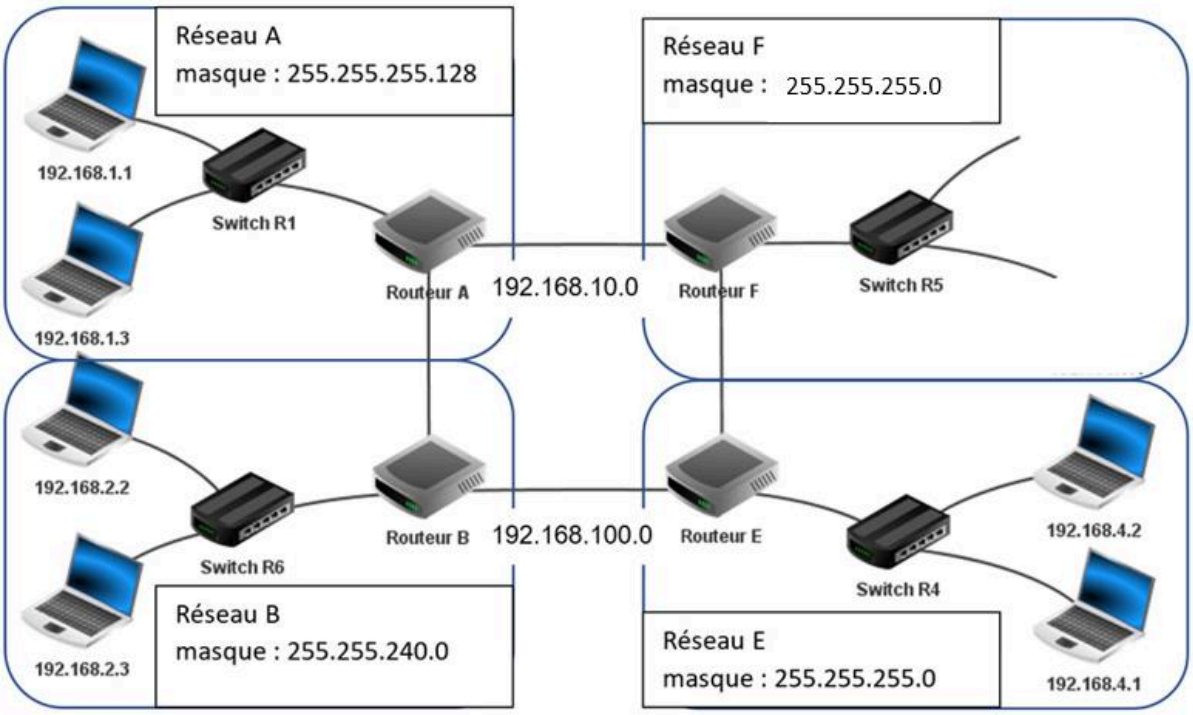


Figure 1. Plan des réseaux étudiés

1. Nous allons considérer le réseau nommé F tel qu'illustré. Son masque de réseau étant, en décimales pointées, 255.255.255.0, les trois premiers octets d'une adresse IP sur ce réseau servent pour la partie réseau de l'adresse (appelée aussi Net ID), le dernier octet sert pour la partie hôte et est propre à chaque machine sur le réseau.
   Une machine connectée au switch R5 possède 192.168.5.3 comme adresse IPV4.
   a. Proposer une adresse IP valide pour le routeur F.
   b. Indiquer le nombre maximum de machines que l'on pourra connecter sur ce réseau F.

2. Pour déterminer la partie d'une adresse IPV4 qui correspond à l'adresse réseau, on effectue un ET logique entre chaque bit de l'adresse IP binaire de l'hôte et celle du masque de sous-réseau. Exemple pour un octet :

	1 1 1 0 1 0 1 0	extrait de l'adresse IP
ET	1 1 1 1 1 0 0 0	extrait du masque du réseau
=	1 1 1 0 1 0 0 0	extrait de l'adresse réseau

On considère le réseau B ;

a. Identifier son masque de sous-réseau sur la figure 1 ci-dessus.

b. Déterminer l'adresse du réseau B, à partir de l'adresse IP d'une machine et du masque de ce réseau. On détaillera soigneusement chaque étape du raisonnement.

c. Proposer un intérêt au fait d'avoir une telle interconnexion entre les quatre routeurs A, B, E et F.

Partie B : Le routage

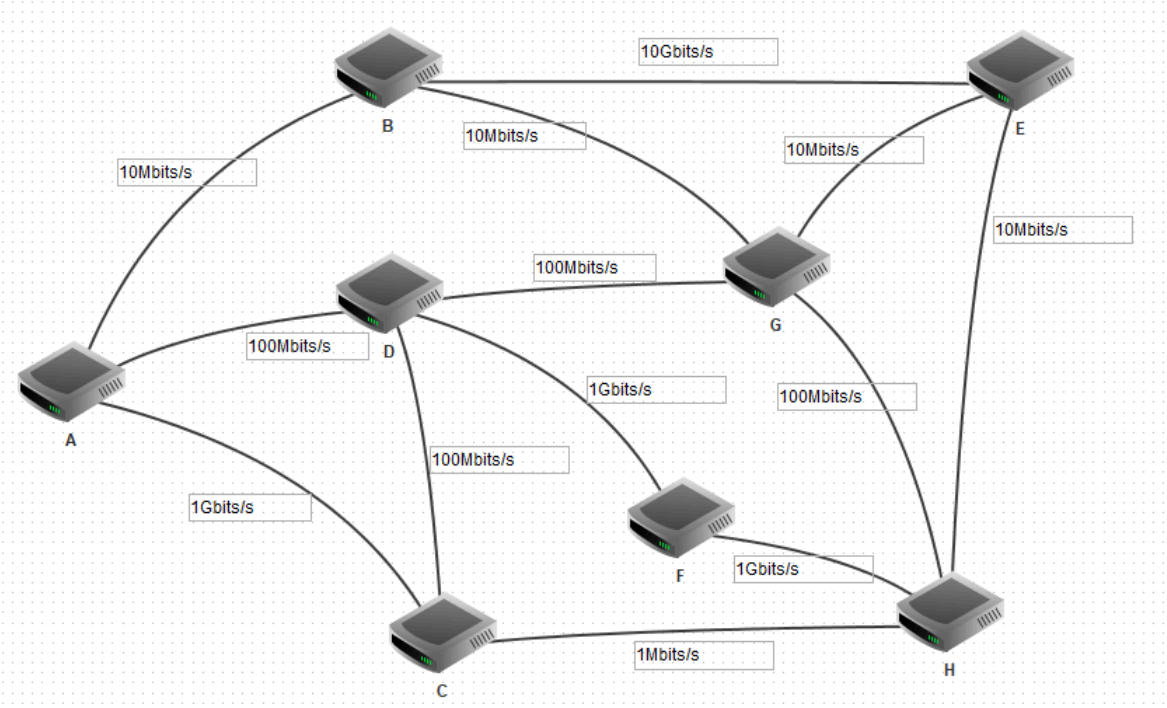


Figure 2. Plan de routage

1. Dans le cadre du protocole RIP, le chemin emprunté par les informations est celui qui aura la distance la plus petite en nombre de sauts. En considérant le réseau présenté ci-dessus :

   a. Donner, en respectant le protocole RIP, le(s) chemin(s) possible(s) entre les routeurs A et E, puis entre les routeurs F et B.

   b. Recopier sur votre copie les tableaux ci-dessous et les compléter pour le routeur E et le routeur G.

Table de routage du routeur E		
Destination	Routeur suivant	Distance
A	B	2
B		
C		
D		
F		
G		
H		

Table de routage du routeur G		
Destination	Routeur suivant	Distance
A		
B		
C		
D		
E		
F		
H		

2. On considère à présent le protocole OSPF qui se base sur le coût total minimal de la communication. Le coût entre deux routeurs se calcule en fonction du débit selon la formule suivante :

$$coût = \frac{10^8}{débit}$$

a. Recopier et compléter la table de routage du routeur F ci dessous.

Table de routage du routeur F		
Destination	Routeur suivant	Coût total
A	D	1,1
B		10,11
C	D	1,1
D	D	
E	H	10,1
G	D	
H	H	0,1

b. Indiquer quel sera le chemin emprunté par les informations entre le routeur E et le routeur D.

EXERCICE 2 (4 points)

Cet exercice porte sur les bases de données relationnelles et le langage SQL.

L'énoncé de cet exercice utilise les mots-clés du langage SQL suivants : `SELECT`, `FROM`, `WHERE`, `JOIN…ON`, `UPDATE…SET`, `INSERT INTO…VALUES…`, `COUNT`, `ORDER BY`.

La clause `ORDER BY` suivie d'un attribut permet de trier les résultats par ordre croissant de l'attribut.

`SELECT COUNT(*)` renvoie le nombre de lignes d'une requête.

Un zoo souhaite pouvoir suivre ses animaux et ses enclos. Tous les représentants d'une espèce sont réunis dans un même enclos. Plusieurs espèces, si elles peuvent cohabiter ensemble, pourront partager le même enclos.

Il crée une base de données utilisant le langage SQL avec une relation (ou table) **`animal`** qui recense chaque animal du zoo. Vous trouverez un extrait de cette relation ci-dessous (les unités des attributs `age`, `taille` et `poids` sont respectivement ans, m et kg) :

animal					
id_animal	nom	age	taille	poids	nom_espece
145	Romy	18	2.3	130	tigre du Bengale
52	Boris	30	1.10	48	bonobo
…	…	…	…	…	…
225	Hervé	10	2.4	130	lama
404	Moris	6	1.70	100	panda
678	Léon	4	0.30	1	varan

Il crée la relation **`enclos`** dont vous trouverez un extrait ci-dessous (l'unité de l'attribut `surface` est m^2) :

enclos				
num_enclos	ecosysteme	surface	struct	date_entretien
40	banquise	50	bassin	04/12/2024
18	forêt tropicale	200	vitré	05/12/2024
…	…	…	…	…
24	savane	300	clôture	04/12/2024
68	désert	2	vivarium	05/12/2024

Il crée également la relation **`espece`** dont vous trouverez un extrait ci-dessous :

espece			
nom_espece	classe	alimentation	num_enclos
impala	mammifères	herbivore	15
ara de Buffon	oiseaux	granivore	77
…	…	…	…
tigre du Bengale	mammifères	carnivore	18
caïman	reptiles	carnivore	45
manchot empereur	oiseaux	carnivore	40
lama	mammifères	herbivore	13

1. Cette question porte sur la lecture et l'écriture de requêtes SQL simples.

   a. Écrire le résultat de la requête ci-dessous.

```
SELECT age, taille, poids FROM animal WHERE nom = 'Moris';
```

   b. Écrire une requête qui permet d'obtenir le nom et l'âge de tous les animaux de l'espèce bonobo, triés du plus jeune au plus vieux.

2. Cette question porte sur le schéma relationnel.

   a. Citer, en justifiant, la clé primaire et la clé étrangère de la relation **`espece`**.

   b. Donner le modèle relationnel de la base de données du zoo. On soulignera les clés primaires et on fera précéder les clés étrangères d'un #.

3. Cette question porte sur les modifications d'une table.

   L'espèce **ornithorynque** a été entrée dans la base comme étant de la **`classe`** des oiseaux alors qu'il s'agit d'un mammifère.

   a. Écrire une requête qui corrige cette erreur dans la table **`espece`**.

   Le couple de lamas du zoo vient de donner naissance au petit lama nommé "Serge" qui mesure 80 cm et pèse 30 kg.

   b. Écrire une requête qui permet d'enregistrer ce nouveau venu au zoo dans la base de données, sachant que les clés primaires de 1 à 178 sont déjà utilisées.

4. Cette question porte sur la jointure entre deux tables

   a. Recopier sur votre feuille la requête SQL et compléter les `...` afin de recenser le nom et l'espèce de tous les animaux carnivores vivant en vivarium dans le zoo.

```
SELECT ...
FROM animal
JOIN espece ON ...
JOIN enclos ON ...
WHERE enclos.struct = 'vivarium' and ...;
```

   On souhaite connaître le nombre d'animaux dans le zoo qui font partie de la classe des **`oiseaux`**.

   b. Écrire la requête qui permet de compter le nombre d'oiseaux dans tout le zoo.

EXERCICE 3 (4 points)

Cet exercice porte sur la programmation en Python, la manipulation des chaînes de caractères, les arbres binaires de recherche et le parcours de liste.

1. On rappelle ici quelques notions sur la manipulation des chaînes de caractères en Python.

Une chaîne de caractères se comporte comme un tableau de caractères que l'on ne peut pas modifier.

Par exemple, on a le comportement suivant :

```
>>> une_chaine = 'Bonjour'
>>> une_chaine[3]
'j'
>>> une_chaine[3] = 'z'
 TypeError: 'str' object does not support item assignment
```

On peut aussi utiliser l'opérateur de concaténation +.

```
>>> une_chaine = 'a' + 'b'
>>> une_chaine
'ab'
>>> une_chaine = une_chaine + 'c'
>>> une_chaine
'abc'
```

On définit la fonction `bonjour` par le code suivant :

```
1  def bonjour(nom)
2      return 'Bonjour ' + nom + ' !'
```

a. Donner le résultat de l'exécution de `bonjour('Alan')`.

On exécute le programme suivant :

```
une_chaine='Bonjour'
x = (une_chaine[2] == une_chaine[3])
y = (une_chaine[4] == une_chaine[1])
```

b. Donner le type et les valeurs des variables `x` et `y`.

c. Écrire une fonction `occurrences_lettre(une_chaine,une_lettre)` prenant en paramètres une chaîne `une_chaine` et une lettre `une_lettre` et renvoyant le nombre d'occurrences de `une_lettre` dans `une_chaine`.

2. On rappelle qu'un arbre binaire de recherche est un arbre binaire pour lequel chaque nœud possède une étiquette dont la valeur est supérieure ou égale à toutes les étiquettes des nœuds de son fils gauche et strictement inférieure à celles des nœuds de son fils droit.
Sa taille est son nombre de nœuds ; sa hauteur est le nombre de niveaux qu'il contient.
On rappelle aussi que l'on peut comparer des chaînes de caractères en utilisant l'ordre alphabétique. On a par exemple :

```
>>> 'ab' < 'aa'
False
>>> 'abc' < 'acb'
True
```

On considère la liste de mots `animaux = ['python', 'chameau', 'pingouin', 'renard', 'gnou']`

a. Dessiner un arbre binaire de recherche contenant tous les mots de la liste `animaux` et de hauteur minimale.

b. Dessiner un arbre binaire de recherche contenant tous ces mots et de hauteur maximale.

3. On considère l'implémentation objet suivante d'un arbre binaire de recherche :
On dispose d'une classe `Abr` contenant notamment les méthodes et attributs suivants :

- Si `un_abr` est une instance d'`Abr` alors `un_abr.est_vide()` renvoie `True` si l'arbre est vide et `False` sinon.
- Si `un_abr` est une instance d'`Abr` et si `un_abr.est_vide()` renvoie `False`, alors `un_abr.valeur` contient une chaîne de caractères représentant la valeur de la racine de l'arbre.
- Si `un_abr` est une instance d'`Abr` et si `un_abr.est_vide()` renvoie `False`, alors `un_abr.sous_arbre_gauche` et `un_abr.sous_arbre_droit` contiennent chacun une instance d'`Abr`.

On considère que la variable `liste_mots_francais` est une liste de 336531 mots en français et que la variable `abr_mots_francais` est une instance d'`Abr` contenant les mots de la liste.

On considère la fonction suivante :

```
1  def mystere(un_abr):
2      if un_abr.est_vide():
3          return 0
4      else:
5          return 1 + mystere(un_abr.sous_arbre_gauche) \
6              + mystere(un_abr.sous_arbre_droit)
```

<u>Remarque</u> : on rappelle que le caractère \ en fin de ligne 5 indique que l'instruction se poursuit sur la ligne suivante.

a. Donner le résultat de `mystere(abr_mots_francais)`, en justifiant le résultat.

On veut calculer la hauteur de `abr_mots_francais`.

b. Donner le code d'une fonction `hauteur(un_abr)` permettant de faire ce calcul, en vous inspirant du code précédent.

4. Dans cette question, nous nous servirons uniquement de `liste_mots_francais` et plus de `abr_mots_francais`.

Pour aider à la résolution de mots croisés, on a décidé d'écrire une fonction `chercher_mots(liste_mots, longueur, lettre, position)` où `liste_mots` est une liste de mots français, `longueur` est la taille du mot recherché, `lettre` est une lettre du mot se trouvant à l'indice `position`.

Par exemple `chercher_mots(liste_mots_francais,3,'x',2)` renverra `['aux', 'box', 'dix', 'eux', 'fax', 'fox', 'lux', 'max', 'six']`.

a. Recopier et compléter la ligne 4 de la fonction ci-dessous :

```
1  def chercher_mots(liste_mots,longueur,lettre,position):
2      res = []
3      for i in range(len(liste_mots)):
4          if ..... and ..... :
5              res.append(liste_mots[i])
6      return res
```

b. Expliquer ce que donne la commande suivante.

```
>>> chercher_mots(chercher_mots(liste_mots_francais,3,'x',2),3,'a',1)
```

On cherche un mot de 5 lettres dont on connait la fin `'ter'`.

c. Ecrire la commande permettant de chercher les mots candidats dans `liste_mots_francais`.