Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2022

Sujet de l'examen national tel que publié pour la session indiquée. Source : sujets nationaux publiés par le ministère chargé de l'examen et les services académiques. Document mis à disposition par FormaV pour la révision.

Toutes les annales et leurs corrigés : https://bac-maths-nsi.fr/annales-sujets-et-corriges

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2022

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé.

Le sujet comporte **16** pages numérotées de **1/16** à **16/16**
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

ATTENTION : Le document réponse (page 16/16) est à rendre obligatoirement avec la copie.

Le candidat traite au choix 3 exercices parmi les 5 exercices proposés.

Chaque exercice est noté sur 4 points.

Description des exercices

Cette page décrit sommairement les connaissances abordées dans chacun des exercices de ce sujet. Le candidat traite au choix 3 exercices parmi les 5 exercices proposés.

EXERCICE 1

Cet exercice porte sur de l'algorithmique et de la programmation en langage Python. Il aborde la programmation orientée objet.

EXERCICE 2

Cet exercice porte sur de l'algorithmique et de la programmation en langage Python utilisant les structures de données du type arbre binaire.

EXERCICE 3

Cet exercice porte sur un schéma relationnel de bases de données et des requêtes SQL.

EXERCICE 4

Cet exercice porte sur l'architecture réseau et des protocoles de communication.

EXERCICE 5

Cet exercice porte sur l'architecture réseau et les tables de routage. Il comporte également l'écriture d'une fonction en langage Python.

EXERCICE 1

Cet exercice porte sur de l'algorithmique et de la programmation en langage Python. Il aborde la programmation orientée objet.

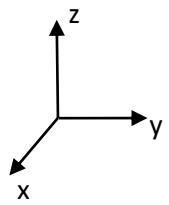
L'objectif de cet exercice est de créer un jeu vidéo. Il s'agit d'un jeu de plateau sur le thème des chevaliers de la table ronde dans lequel plusieurs personnages doivent réaliser des missions.

Ces personnages doivent récupérer des objets sur leur parcours. La récupération de ces objets leur permettra de gagner des points de vie ou d'en perdre.

Dans l'environnement du jeu, les héros vont rencontrer des personnages qui pourraient se révéler dangereux et contre lesquels ils devront parfois engager des batailles.

Partie 1

On s'intéresse ici au déplacement des personnages. L'espace dans lequel ils évoluent est représenté par un repère orthonormé à trois axes. La position de chaque personnage sera repérée par ses attributs x, y, z.



1)

a) **Recopier** sur la copie et **compléter** le constructeur de la classe `Personnage` positionnant un personnage aux coordonnées choisies x, y et z.

```
class Personnage:
    def __init__(self, coordx, coordy, coordz):
            ……
            ……
            ……
```

b) **Écrire** une méthode `avancex` de la classe `Personnage` permettant d'augmenter d'une unité la coordonnée x d'un personnage.

c) **Écrire** une méthode `raz` permettant de mettre toutes les coordonnées d'un personnage à zéro.

d) **Écrire** une méthode `coord` renvoyant les coordonnées d'un personnage sous forme d'un tuple.

2) En utilisant les méthodes écrites dans la partie 1, **écrire** des lignes de code permettant :

a) de créer un personnage `arthur` démarrant à la position (5,5,5) ;

b) d'augmenter d'une unité la coordonnée x du personnage `arthur` ;

c) d'afficher les coordonnées du personnage `arthur`.

Partie 2

Chaque personnage du jeu va rencontrer différentes situations (potions, pièges, …) qui vont faire évoluer ses points de vie. La classe `Personnage` est modifiée afin de permettre ces évolutions.

```
import random
class Personnage :
     def __init__(self, coordx, coordy, coordz, point_de_vie):
          ... # défini dans la partie 1
          self.vie = point_de_vie
     def avancex ... # défini dans la partie 1
     def raz ... # défini dans la partie 1
     def coord ... # défini dans la partie 1

     def get_etat(self) :
          return self.vie
     def newgame(self) :
          ……   # défini dans la partie 2
     def potionmystere(self) :
          if random.randint(1,2) == 1 :
               nbPoint = -1
          else :
               nbPoint = +1
          self.vie = self.vie + nbPoint
     def piege(self) :
          self.vie = self.vie - 10
     def repos(self) :
          self.vie = self.vie + 5
```

1) **Indiquer** les valeurs possibles de la nouvelle variable `valeurMerlin` après exécution du programme ci-dessous.

```
merlin = Personnage (4,5,8,15)
merlin.potionmystere()
valeurMerlin = merlin.get_etat()
```

2) **Indiquer** la valeur de la nouvelle variable `valeurMerlin` après exécution du programme ci-dessous.

```
merlin = Personnage (4,5,8,20)
merlin.piege()
merlin.piege()
valeurMerlin = merlin.get_etat()
```

3) **Écrire** sur la copie, la méthode `newgame` permettant, si le nombre de points de vie est inférieur ou égal à 0, de :

   - ramener les coordonnées du personnage à (0,0,0),
   - lui attribuer 15 points de vie.

Pour intégrer les combats au jeu, on modifie et complète la classe `Personnage` en rajoutant les méthodes suivantes :

```
def perdre_vie(self,points) :
     self.vie = self.vie - points
     self.newgame()
def attaquer(self, autre) :
     autre.perdre_vie(self.degats)
```

4) **Écrire** un programme en langage Python permettant :

   a) la création d'une instance `lancelot` de cette classe ayant pour attributs 5,5,5 en coordonnées, 15 en `point_de_vie` et 3 en `point_degats`.

   b) la création d'une instance `sorcier` de cette classe ayant pour attributs 6,5,5 en coordonnées, 15 en `point_de_vie` et 2 en `point_degats`.

   c) à `lancelot` d'attaquer le `sorcier` une première fois.

   d) au `sorcier` d'attaquer à son tour `lancelot`.

   e) à `lancelot` de répliquer en attaquant quatre fois de suite le `sorcier`.

   f) d'afficher le nombre de points de vie de `lancelot` et du `sorcier`.

EXERCICE 2

Cet exercice porte sur de l'algorithmique et de la programmation en langage Python utilisant les structures de données du type arbre binaire.

On crée un jeu dans lequel les personnages doivent se déplacer. Le personnage Mélusine se déplace et se retrouve devant différents chemins possibles. Le schéma 1 ci-dessous illustre la situation.

Mélusine part du point A et peut se rendre à n'importe quel autre point repéré par une lettre. Sous chaque lettre, se cache un objet à récupérer. Chaque objet est associé à une valeur.

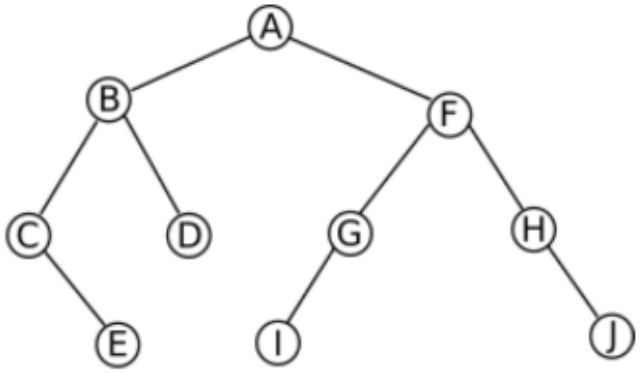


Schéma 1

1) **Indiquer** pourquoi il s'agit d'un arbre binaire.

2) À chaque lettre, on attribue une valeur positive. On définit la variable `V` suivante contenant les valeurs associées aux lettres :

```
V = {'A': 1 , 'B': 2 , 'C': 3 , 'D': 5 , 'E': 10 , 'F': 15 , 'G': 4 , 'H': 5 , 'I': 5 , 'J': 7}
```

   a) **Indiquer** le type de la variable V.

   b) **Écrire** l'instruction permettant d'accéder à la valeur « 7 » stockée dans cette variable.

   c) **Écrire** une fonction `somme` en langage Python qui prend un paramètre `W` du même type que la variable `V` et qui renvoie la somme des valeurs de `W`.

   d) **Écrire** une fonction `VMax` en langage Python qui prend un paramètre `W` du même type que la variable `V` et qui renvoie la lettre ayant la valeur maximale de `W`.

3) **Indiquer**, en **justifiant** la réponse, le rôle de l'algorithme suivant.

```
CALCUL(T : arbre) :
VARIABLE
x : noeud
DEBUT
  si T n'est pas un arbre vide :
    x ← racine de T
    renvoyer 1 + CALCUL(sous arbre gauche de x) + CALCUL(sous arbre
droit de x)
  sinon :
    renvoyer 0
  fin si
FIN
```

4) On applique l'algorithme ci-dessous à l'arbre du schéma 1.

```
VISITE(T: arbre) :
VARIABLE
x : noeud
DEBUT
  si T n'est pas un arbre vide :
    x ← racine de T
    affiche clé de x
    VISITE(sous arbre gauche de x)
    VISITE(sous arbre droit de x)
  fin si
FIN
```

a) **Indiquer** l'affichage obtenu.

b) **Indiquer** le type de parcours réalisé par l'algorithme `VISITE`.

EXERCICE 3

Cet exercice porte sur un schéma relationnel de bases de données et des requêtes SQL.

Pour élaborer un jeu vidéo sur le thème des chevaliers de la table ronde, on crée une base de données **`Chevalier`**. Celle-ci permet de disposer de toutes les caractéristiques de tous les personnages du jeu. Deux tables sont créées avec différents attributs.

Table Personnage

Idperso	nom	frere_de	points	Idqualite
1	Merlin	NC	40	2
2	Galehaut	NC	40	5
3	Lancelot	Hector	50	1
4	Gareth	Gaheris	20	4
5	Galahad	NC	25	3
6	Gaheris	Gareth	30	1
7	Erek	NC	25	1
8	Lamorak	Perceval	30	1
9	Perceval	Lamorak	35	3
10	Hector	Lancelot	50	6
11	Keud	Arthur	80	7
12	Bedivere	Lucan	20	6
13	Lucan	Bedivere	25	7

Table Qualite

Idqualite	nom_qualite
1	apprenti chevalier
2	magicien
3	preux chevalier
4	espion
5	seigneur
6	sage
7	grand chevalier

NC signifie : « Non communiqué »
Dans la suite, toutes les requêtes seront écrites en langage SQL.

1) **Écrire** une requête permettant d'afficher le nom et les points de tous les personnages.

2) Une erreur a été identifiée. Le frère d'Arthur ne s'appelle pas Keud mais Antor.
Écrire une requête permettant la modification de ce nom.

3) **Écrire** une requête permettant d'ajouter le `nom_qualite` « roi » à la table `Qualite` ayant 8 pour `Idqualite`.

4) **Écrire** une requête permettant de créer un quatorzième personnage : le roi Arthur avec 100 points.

5) **Écrire** une requête permettant d'afficher le nom et le nom de la qualité des personnages dont les points sont égaux à 40.

6) On applique la requête suivante :
UPDATE Personnage **SET** points=points+10
WHERE points < 40;

Indiquer alors le nombre de points des personnages suivants : Arthur, Perceval et Merlin.

EXERCICE 4

Cet exercice porte sur l'architecture réseau et des protocoles de communication.

On souhaite tester un jeu vidéo en mettant en place un réseau d'ordinateurs répartis dans trois salles.

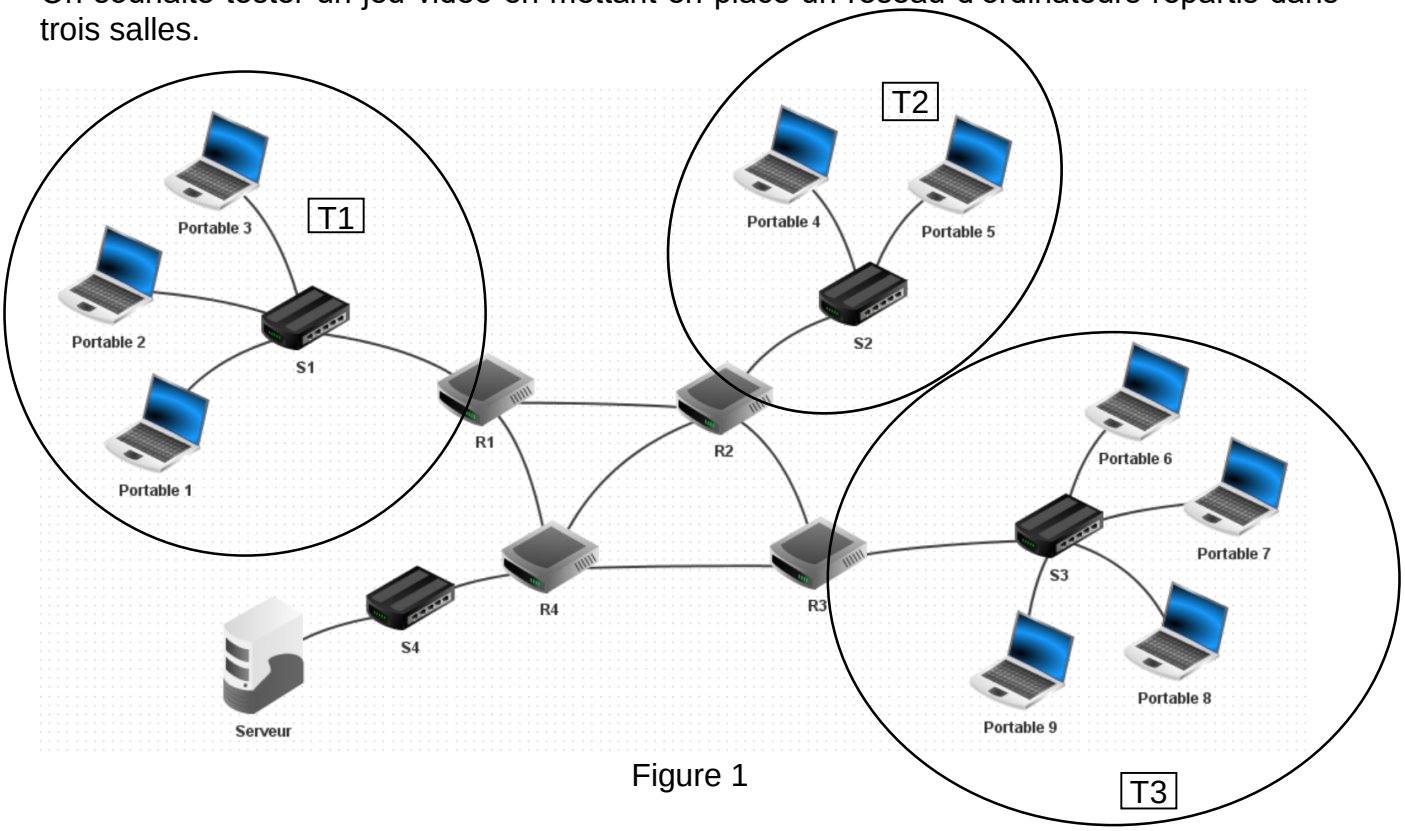


Figure 1

La figure 1 représente le schéma du réseau d'ordinateurs étudié. Il comprend trois réseaux locaux T1, T2 et T3 dans lesquels vont s'installer les joueurs. Un quatrième réseau local sera utilisé pour un serveur de jeux.

Ces réseaux locaux sont interconnectés grâce à des routeurs R1, R2, R3 et R4. Les réseaux locaux T1, T2 et T3 sont constitués de plusieurs ordinateurs portables nommés « Portable i », $1 \leq i \leq 9$, et de commutateurs (switchs) nommés S1, S2 et S3.

Le serveur est connecté au routeur (passerelle) R4 par l'intermédiaire du commutateur (switch) S4.

Rappels et notations :
Une adresse IPv4, codée sur 4 octets, doit être associée à un masque de réseau pour être interprétable.

Le masque de réseau :

- permet de distinguer la partie de l'adresse qui identifie un réseau de celle qui identifie une machine ;
- est codé sur 4 octets soit 32 bits sous la forme d'une suite de 1 puis une suite de 0 ;
- peut être indiqué sous forme décimale ou en notation CIDR (Classless Inter-Domain Routing) « /n » où n correspond aux nombres de bits égaux à 1.

Il peut être codé sous la forme 1111 1111.0000 0000.0000 0000.0000 0000 ou en ajoutant « /8 » à l'adresse IP (exemple : 172.16.1.1/8).

Le tableau 1 donne des informations sur les adresses IP de la plupart des éléments constituant le réseau : le nom de l'élément, son type, l'adresse IPv4 de son ou ses interfaces par lesquelles sortent les paquets, l'élément directement relié à l'interface.

Nom	Type	Adresse IP	Côté
R1	routeur	Interface 1 : 195.168.1.1/24 Interface 2 : 196.163.2.1/24 Interface 3 : 194.162.1.1/24	S1 R2 R4
R2	routeur	Interface 1 : 197.162.1.1/24 Interface 2 : 196.163.2.2/24 Interface 3 : 198.164.3.1/24 Interface 4 : 193.154.5.1/24	S2 R1 R3 R4
R3	routeur	Interface 1 : ……..…/24 Interface 2 : ……../24 Interface 3 : ……../24	S3 R2 R4
R4	routeur	Interface 1 : 220.10.1.1/24 Interface 2 : 194.162.1.2/24 Interface 3 : 193.154.5.2/24 Interface 4 : 200.158.4.2/24	S4 R1 R2 R3
Portable 1	ordinateur portable	195.168.1.40/24	S1
Portable 5	ordinateur portable	197.162.1.50/24	S2
Portable 6	ordinateur portable	199.160.1.60/24	S3
Portable 7	ordinateur portable	199.160.1.61/24	S3
Portable 8	ordinateur portable	199.160.1.62/24	S3
Portable 9	ordinateur portable	199.160.1.63/24	S3
Serveur	serveur	220.10.1.12/24	S4

Tableau 1

1) a) **Indiquer** l'adresse du réseau local dont fait partie l'ordinateur Portable 3.
b) **Indiquer** une adresse possible pour l'ordinateur Portable 3.
c) **Indiquer,** en justifiant, le nombre d'adresses encore disponibles pour l'ordinateur Portable 4 du réseau local T2.

2) **Indiquer** les adresses IP des interfaces du routeur 3. Chaque adresse IP est la première disponible sur la plage d'adressage de chaque réseau connecté.

3) Lors du jeu, l'ordinateur Portable 1 veut communiquer avec l'ordinateur Portable 5.

   a) **Indiquer** trois parcours possibles en listant tous les éléments utilisés du réseau.
   b) **Indiquer** le plus court chemin au sens du protocole RIP, minimisant le nombre de routeurs traversés (sauts), en précisant son nombre de sauts.
   c) La liaison R1-R2 vient de se rompre. **Indiquer** le plus court chemin au sens du protocole RIP, minimisant le nombre de routeurs traversés.

4) La liaison R1-R2 entre les deux réseaux locaux est rétablie en changeant le câble de raccordement. Parmi les quatre propositions suivantes, **indiquer** le type de ce câble.

a) Internet	b) VGA	c) Ethernet	d) HDMI

5) On souhaite déterminer le parcours minimisant le coût total des liaisons traversées. Pour cela, le protocole OSPF est utilisé.

   On s'intéresse donc au parcours à moindre coût entre le routeur R1 et le routeur R2. Étant donné une bande passante de référence de 10^9 bps (bits par seconde), le coût d'une liaison entre routeurs est donné par la formule suivante où d est la bande passante en bps entre les deux routeurs.

$$\text{coût} = \frac{10^9}{d}$$

   On admet que si le résultat du quotient précédent est inférieur ou égal à 1, le coût est égal à 1.

On donne dans le tableau ci-dessous les débits des liaisons entre routeurs.
Note : Mbps signifie 10^6 bits par seconde.

Liaison	R1-R2	R1-R4	R2-R3	R2-R4	R3-R4
Débit en Mbps	?	1000	10	1000	20

   a) La liaison entre le routeur R1 et R2 a un coût de 10.
   Calculer le débit de cette liaison en Mbps.

   b) L'ordinateur Portable 1 communique avec l'ordinateur Portable 5. Le routeur R1 doit transmettre les paquets au routeur R2.
   Déterminer le chemin de meilleur coût et **indiquer** ce coût. **Justifier** les réponses.

EXERCICE 5

Cet exercice porte sur l'architecture réseau et les tables de routage. Il comporte également l'écriture d'une fonction en langage Python.

1) Une adresse IPv4 doit être associée à un masque de réseau pour être interprétable.

Rappel sur le masque de réseau :

- il permet de distinguer la partie de l'adresse qui identifie un réseau de celle qui identifie une machine ;
- il est codé sur 4 octets soit 32 bits sous la forme d'une suite de 1 puis une suite de 0 ;
- il peut être indiqué sous forme décimale ou en notation CIDR (Classless Inter-Domain Routing) « /n » où n correspond aux nombres de bits égaux à 1.

Exemples :

- 255.255.192.0 est un masque de réseau valide écrit sous forme décimale et de notation CIDR /18.
  En effet, sa forme binaire contient dix-huit 1 suivis d'une suite de 0.
  1 1 1 1 1 1 1 1 . 1 1 1 1 1 1 1 1 . 1 1 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0

- 255.252.128.0 n'est pas un masque de réseau valide car sa forme binaire n'est pas une suite de 1 suivie d'une suite de 0 : elle contient au moins un 0 intercalé entre deux 1.
  1 1 1 1 1 1 1 1 . 1 1 1 1 **1 1 0 0 . 1** 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0

a) Pour chaque écriture de masque de réseau, ci-dessous, **indiquer** et **justifier** si elle correspond à un masque de réseau valide :

- 255.255.225.0
- 255.255.224.0

b) On dispose d'une fonction `convBin` qui prend en paramètre une liste de quatre nombres entiers compris entre 0 et 255 et qui renvoie la liste des 32 bits de l'écriture binaire correspondante.

Par exemple :

- `convBin( [255,255,192,0] )` renvoie
  `[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]`

- `convBin( [255,252,128,0] )` renvoie
  `[1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]`

Dans cette question, on considère que les masques en notation CIDR /0 et /32 sont valides.
Écrire en langage Python la fonction `cidr` qui prend en paramètre une liste de 32 bits et qui :

- renvoie l'entier n de la notation CIDR si la liste des 32 bits correspond à l'écriture binaire d'un masque de réseau valide ;
- renvoie -1 sinon.

Par exemple :

- m=`convBin`([255,255,192,0])
  `cidr`(m) renvoie 18
- m=`convBin`([255,252,128,0])
  `cidr`(m) renvoie -1

2) Parmi les réponses ci-dessous, **indiquer** celle qui propose des commandes permettant d'afficher la table de routage.

- Réponse A : dir (sous windows) ou ls (sous linux)
- Réponse B : cacls (sous windows) ou chmod (sous linux)
- Réponse C : route print (sous windows) ou ip route (sous linux) ou route – n (sous linux)
- Réponse D : ping
- Réponse E : tracert (sous windows) ou traceroute (sous linux)

3) Le réseau étudié, constitué de 3 routeurs R1, R2 et R3 est représenté à la figure 1. Chaque routeur possède deux interfaces eth0 et eth1 dont les adresses IPv4 sont indiquées ci-dessous.

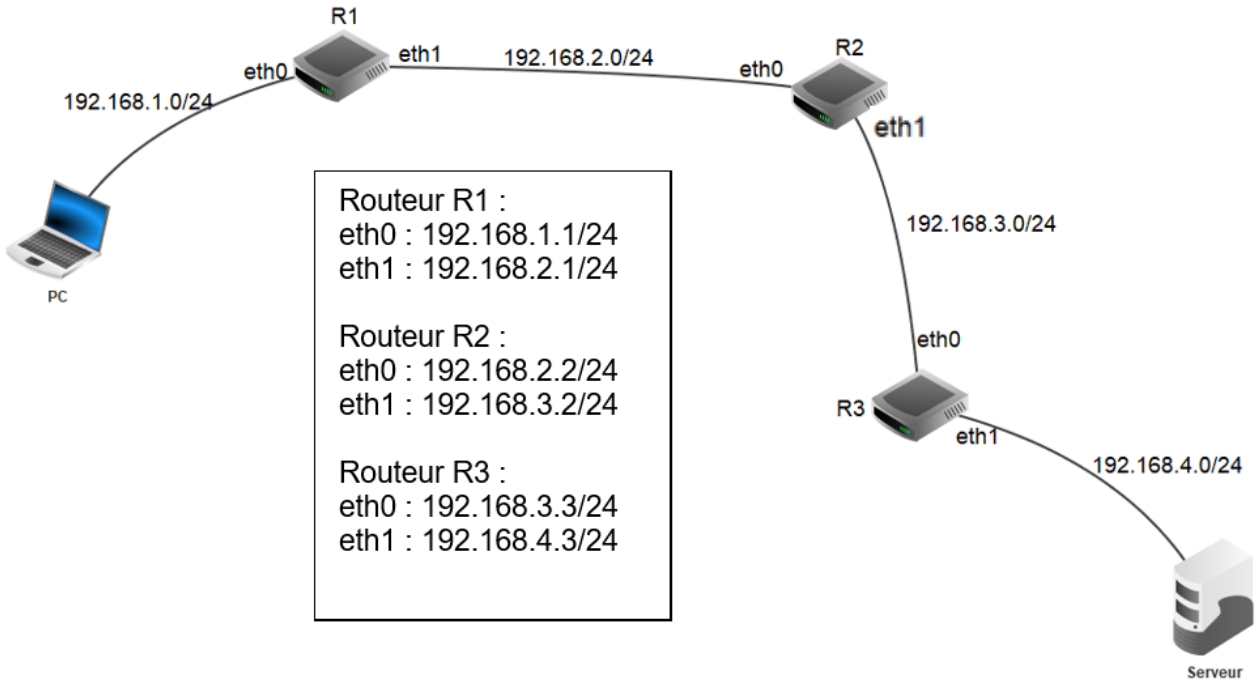


Figure 1

La table de routage d'un routeur est constituée des informations suivantes :

- l'adresse IP de destination ;
- l'adresse IP de l'interface du routeur par lequel **sort** le paquet ;
- dans la colonne « Routeur suivant », si le routeur est directement connecté au réseau alors on note « direct », sinon on inscrit l'adresse IP de l'interface du routeur suivant par lequel **doit entrer** le paquet ;
- le nombre de routeurs (sauts) traversés pour atteindre la destination.

a) On appelle étape n°1, l'étape correspondant à la mise en service des routeurs. Au départ, les routeurs connaissent uniquement les adresses réseaux de leurs routeurs voisins.

À l'étape n°1, la table de routage du routeur R1 est la suivante :

Routeur R1			
Destination	Interface	Routeur suivant	Saut
192.168.1.0/24	192.168.1.1	direct	0
192.168.2.0/24	192.168.2.1	direct	0

Compléter, sur le document réponse Exercice 5 page 16, les tables de routage des routeurs R2 et R3 à l'étape n°1.

b) À l'étape n°2, les routeurs communiquent dans l'ordre suivant :

- R2 envoie le contenu de sa table à ses voisins puis les routeurs R1 et R3 mettent à jour leur table ;
- R1 et R3 envoient le contenu de leur table à R2, puis le routeur R2 met à jour sa table de routage.

Compléter, sur le document réponse Exercice 5 page 16, la table de routage du routeur R1 à l'étape n°2.

La table de routage de R2 se stabilisant à partir de l'étape n°2 est :

Routeur R2			
Destination	Interface	Routeur suivant	Saut
192.168.2.0/24	192.168.2.2	direct	0
192.168.3.0/24	192.168.3.2	direct	0
192.168.1.0/24	192.168.2.2	192.168.2.1	1
192.168.4.0/24	192.168.3.2	192.168.3.3	1

c) À l'étape n°3, les routeurs communiquent dans l'ordre suivant :

- R2 envoie le contenu de sa table à ses voisins puis les routeurs R1 et R3 mettent à jour leur table ;
- R1 et R3 envoient le contenu de leur table à R2, puis le routeur R2 met à jour sa table de routage.

Compléter, sur le document réponse Exercice 5 page 16, la table de routage de R1 à l'étape n°3.

Document réponse : EXERCICE 5

(compléter le verso de cette page)

3) a)

Routeur 2			
Destination	Interface	Routeur suivant	Saut

Routeur 3			
Destination	Interface	Routeur suivant	Saut

3) b)

Routeur 1			
Destination	Interface	Routeur suivant	Saut

3) c)

Routeur 1			
Destination	Interface	Routeur suivant	Saut

Modèle CCYC : ©DNE

NOM DE FAMILLE (naissance) :
(en majuscules)

PRENOM :
(en majuscules)

N° candidat :

N° d'inscription :

(Les numéros figurent sur la convocation, si besoin demander à un surveillant.)

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

1.2