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

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2023

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

Centres étrangers - mars 2023 - sujet 1

Exercice 1 (Bases de données et SQL - 3 points)
L'énoncé de cet exercice utilise les mots clefs suivants du langage SQL : `SELECT`, `FROM`, `WHERE`, `JOIN ... ON`, `UPDATE ... SET`, `DELETE`, `INSERT INTO ... VALUES`, `ORDER BY`.

On rappelle que la clause `ORDER BY` suivie d'un attribut permet de trier les résultats par ordre croissant des valeurs de l'attribut.

Radio France souhaite créer une base de données relationnelles contenant les podcasts des émissions de radio. Pour cela, elle utilise le langage SQL. Elle crée :

- une relation (ou table) `podcast` qui contient le thème et l'année de diffusion ;
- une relation `emission` qui contient les émissions (`id_emission`, `nom`), la radio de diffusion et l'animateur ;
- une relation `description` qui contient un résumé et la durée du podcast en minutes.

Voici un extrait de ces relations :

- Relation `podcast`

id_podcast	**theme**	**annee**	**id_emission**
1	Le système d'enseignement supérieur français est-il juste et efficace ?	2022	10081
2	Trois innovations pour la croissance future (1/3) : la révolution blockchain	2021	10081
3	Travailleurs de plateformes : vers un nouveau prolétariat ?	2021	10175
4	Le poids de la souveraineté numérique française	2019	10183
40	Le poids de la souveraineté numérique française	2019	10183
5	Dans le cloud en Islande, terre des data center	2019	10212

- Relation `emission`

id_emission	**nom**	**radio**	**animateur**
10081	Entendez-vous l'éco ?	France culture	Tiphaine de R.
10175	Le temps du débat	France culture	Léa S.
10183	Soft power	France culture	Frédéric M.
10212	La tête au carré	France inter	Mathieu V.

`id_podcast` de la relation `podcast` et `id_emission` de la relation `emission` sont des clés primaires.
L'attribut `id_emission` de la relation `podcast` fait directement référence à la clé primaire de la relation `emission`.

⋆ Relation `description`

id_description	resume	duree	id_emission
101	Autrefois réservé à une élite, l'enseignement supérieur français s'est profondément démocratisé : donne-t-il pour autant les mêmes chances à chacun ?	4	10081
102	Quelles sont leurs conditions de travail et quels sont leurs moyens de contestation ?	58	10175
103	La promesse de la blockchain, c'est la suppression des intermédiaires et la confiance à grande échelle	4	10081

1. Ecrire le schéma relationnel de la relation `description` en précisant les attributs et leurs types probables, la clé primaire et la ou les clé(s) étrangère(s) éventuelle(s).
2. (a) Ecrire ce qu'affiche la requête suivante appliquée aux extraits précédents :

```
SELECT theme, annee FROM podcast WHERE id_emission = 10081
```

   (b) Ecrire une requête SQL permettant d'afficher les thèmes des podcasts de l'année 2019.
   (c) Ecrire une requête SQL affichant la liste des thèmes et des années de diffusion des podcasts dans l'ordre chronologiques des années.
3. (a) Décrire simplement le résultat obtenu avec cette requête SQL :

```
SELECT DISTINCT theme FROM podcast
```

   (b) Ecrire une requête SQL supprimant la ligne contenant l'`id_podcast = 40` de la relation `podcast`.
4. (a) Une erreur de saisie a été faite dans la relation `emission`. Ecrire une requête SQL permettant de changer le nom de l'animateur de l'émission `Le temps du débat` en `Emmanuel L.`.
   (b) Ecrire une requête SQL permettant d'ajouter l'émission `Hashtag` sur la radio `France inter` avec `Mathieu V.`. On lui donnera un `id_emission` égal à `12850`.
5. Ecrire une requête permettant de lister les thèmes, le nom des émissions et le résumé des podcasts pour lesquels la durée est strictement inférieure à cinq minutes.

Exercice 2 (Réseaux et protocoles de routage - 3 points)
Voici ci-dessous un réseau dans lequel A, B, C, D, E, F, G et H sont des routeurs.

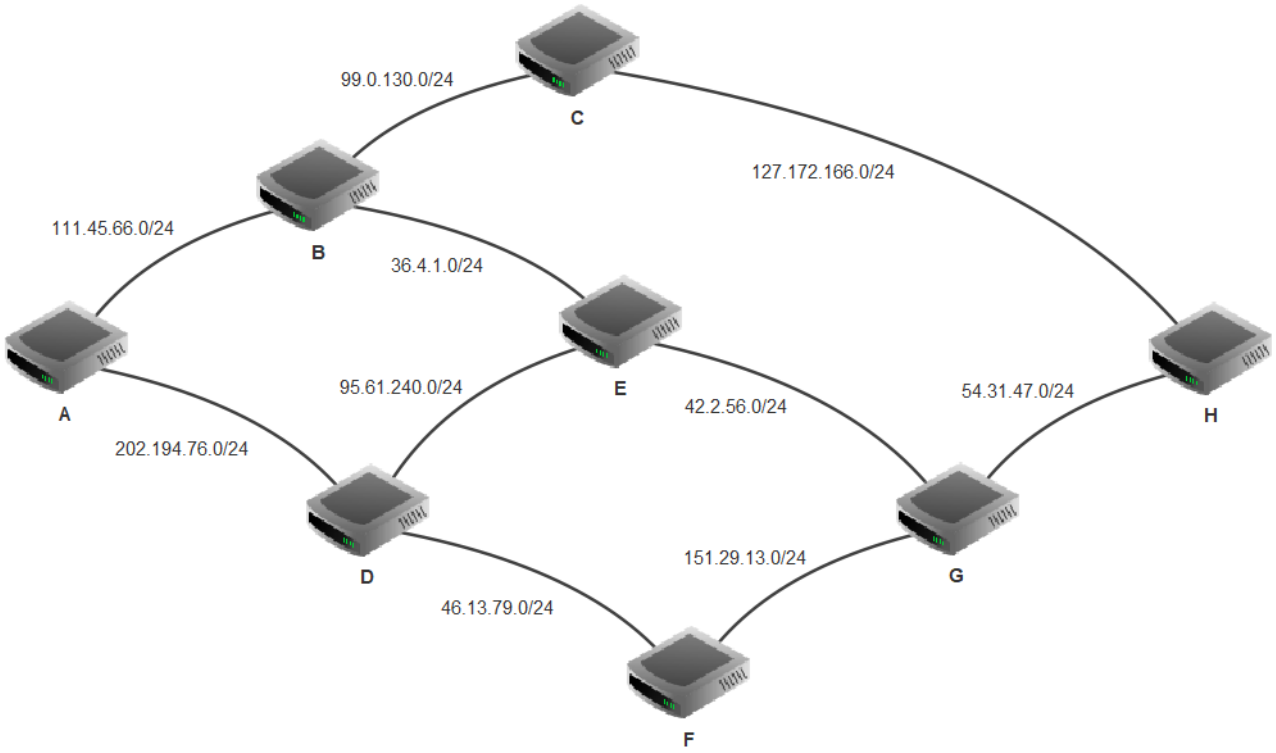


FIGURE 1 – Réseau de routeurs

Les adresses IP seront conformes à la norme IPV4, à savoir composées de quatre octets. Elles prendront la forme `X1.X2.X3.X4`, où `X1`, `X2`, `X3` et `X4` sont les valeurs des quatre octets convertis en notation décimale.

La notation CIDR `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 représentent la partie « hôte ».

Toutes les adresses hôtes connectés à un réseau local ont la même partie réseau et peuvent donc communiquer directement. L'adresse IP dont tous les bits de la partie « hôte » sont à `0` est appelée « adresse du réseau ».

1. (a) `10100100.10110010.XXXXXXXX.XXXXXXXX` est la conversion en binaire de l'adresse `164.178.2.13`. Terminer cette conversion en remplaçant les deux octets `XXXXXXXX` par leur valeur binaire.
   (b) Donner, en justifiant, l'adresse du réseau à laquelle appartient la machine dont l'adresse complète en notation CIDR est `164.178.2.13/24`.

Le protocole RIP (Routing Information Protocol) est un protocole de routage qui cherche à minimiser le nombre de routeurs traversés (ce qui correspond à la distance ou au nombre de sauts).

2. Donner tous les chemins de parcours **optimaux** pour un paquet émis par A et à destination de G en suivant le protocole RIP.

Voici le réseau de la figure 1 indiquant le type de connexion entre les routeurs.

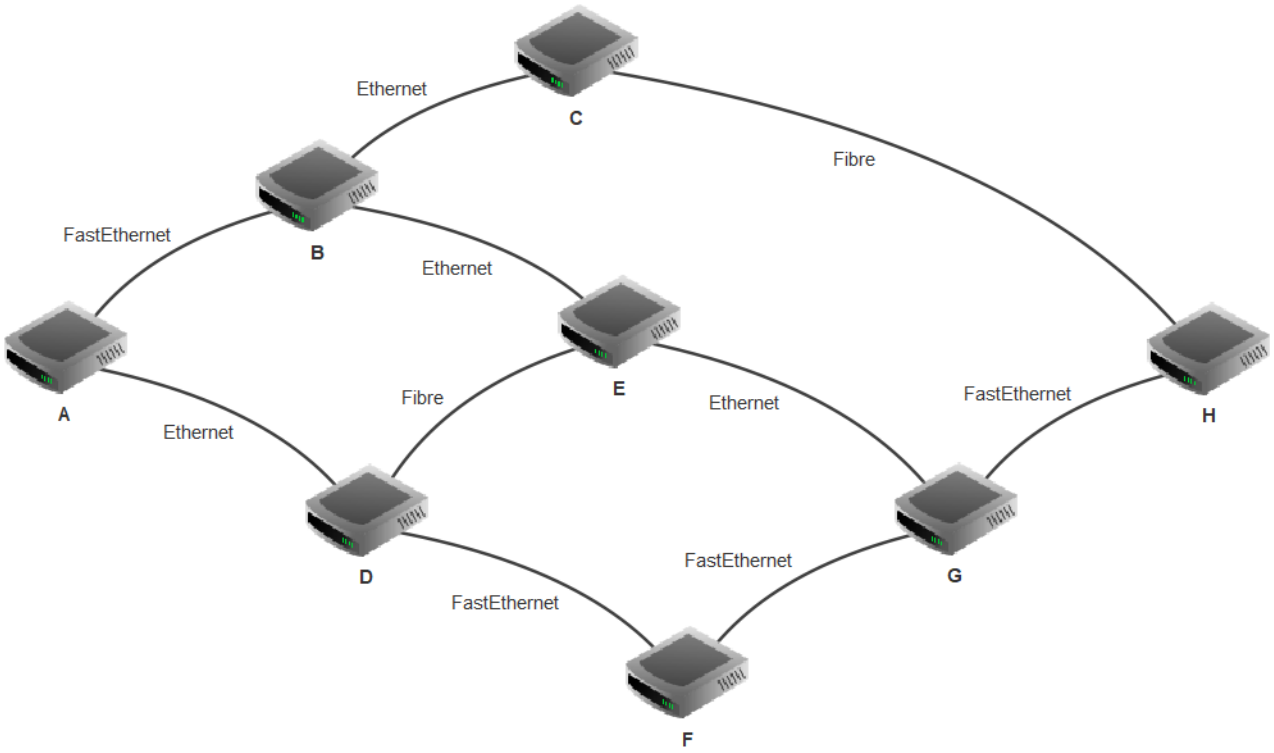


FIGURE 2 – Réseau de routeurs avec les types de connexion

Nous allons travailler avec le protocole de distance en coût des routes (OSPF).
On considère le coût d'une liaison en fonction du type de connexion donné par la formule :

$$\text{coût} = \frac{10^9}{\text{BP}},$$

où BP est la bande passante en bit/s.

Connexion	**BP estimée en bit/s**
Ethernet	10^8
FastEthernet	10^9
Fibre	10^{10}

3. (a) Dessiner sur votre copie le schéma du réseau en remplaçant le type de connexion par le coût. On se limitera aux noms des routeurs et aux coûts.
   (b) Donner le chemin de parcours pour un paquet émis par A et à destination de G en respectant le protocole OSPF.
   (c) Donner le chemin de parcours pour un paquet émis par A et à destination de G en respectant le protocole OSPF si le routeur F est en panne.

Exercice 3 (Files - 6 points)

Simon *est un jeu de société électronique de forme circulaire comportant quatre grosses touches de couleurs différentes : rouge, vert, bleu et jaune.*
Le jeu joue une séquence de couleurs que le joueur doit mémoriser et répéter ensuite. S'il réussit, une couleur parmi les quatre est ajoutée à la fin de la séquence. La nouvelle séquence est jouée depuis le début et le jeu continue. Dès que le joueur se trompe, la séquence est vidée et réinitialisée avec une couleur et une nouvelle partie recommence.
Exemple de séquence jouée : rouge → bleu → rouge → jaune → bleu. Source Wikipédia

Dans cet exercice, nous essaierons de reproduire ce jeu.

- Les quatre couleurs sont stockées dans un tuple nommé `couleur` :

```
couleurs = ('bleu', 'rouge', 'jaune', 'vert')
```

- Pour stocker la séquence à afficher, nous utiliserons une structure de file que l'on nommera `sequence` tout au long de l'exercice.

La file est une structure linéairede type FIFO (First In First Out). Nous utiliserons durant cet exercice les fonctions suivantes :

Structure de données abstraites : File	
`creer_file_vide()`	: renvoie une file vide
`est_vide(f)`	: renvoie `True` si `f` est vide, `False` sinon
`enfiler(f, element)`	: ajoute `element` en queue de `f`
`defiler(f)`	: retire l'élément en tête de `f` et le renvoie
`taille(f)`	: renvoie le nombre d'éléments de `f`

En fin de chaque séquence, le Simon tire au hasard une couleur parmi les quatre proposées. On utilisera la fonction `randint(a, b)` de la bibliothèque `random` qui permet d'obtenir un nombre entier compris entre `a` inclus et `b` inclus pour le tirage aléatoire.
Exemple : `randint(1, 5)` peut renvoyer `1`, `2`, `3`, `4` ou `5`.

1. Recopier et compléter sur votre copie les pointillés des lignes 3 et 4 de la fonction `ajout(f)` qui permet de tirer au hasard une couleur et de l'ajouter à une séquence. La fonction `ajout` prend en paramètre la séquence `f` et renvoie la séquence `f` modifiée (qui intègre la couleur ajoutée au format chaîne de caractères).

```
1 def ajout(f):
2     couleur = ('bleu', 'rouge', 'jaune', 'vert')
3     indice = randint(....., .....)
4     enfiler(....., .....)
5     return f
```

En cas d'erreur du joueur durant sa réponse, la partie reprend au début. Il faut donc vider la file `sequence` pour recommencer à zéro en appelant `vider(sequence)` qui permet de rendre la file `sequence` vide sans la renvoyer.

2. Ecrire la fonction `vider` qui prend en paramètre une séquence `f` et la vide sans la renvoyer.

Le Simon doit afficher successivement les différentes couleurs de la séquence. Ce rôle est confié à la fonction `affiche_seq(sequence)` qui prend en paramètre la file de couleurs `sequence` et qui est définie par l'algorithme suivant :

- on ajoute une nouvelle couleur à `sequence` ;
- on affiche les couleurs de la séquence, une par une, avec une pause de $0{,}5$ seconde entre chaque affichage.

Une fonction `affichage(couleur)`, dont la rédaction n'est pas demandée dans cet exercice, permettra l'affichage de la couleur souhaitée avec `couleur` de type chaîne de caractères correspondant à une des quatre couleurs.
La temporisation de $0{,}5$ seconde sera effectuée avec la commande `time.sleep(0.5)`.
Après l'exécution de la fonction `affiche_seq`, la file `sequence` ne devra pas être vidée de ses éléments.

3. Recopier et compléter sur votre copie les pointillés des lignes 4 à 10 de la fonction `affiche_seq(sequence)` ci-dessous.

```
1  def affiche_sequence(sequence):
2      stock = creer_file_vide()
3      ajout(sequence)
4      while not est_vide(sequence):
5          c = .....
6          .....
7          time.sleep(0.5)
8          .....
9      while ..... :
10         .....
```

4. *Cette question est indépendante des précédentes : bien qu'elle fasse appel aux fonctions construites précédemment, elle peut être résolue même si le candidat n'a pas réussi toutes les questions précédentes.*
Nous allons créer ici une fonction `tour_de_jeu(sequence)` qui gère le déroulement d'un tour quelconque de jeu côté joueur.
La fonction `tour_de_jeu` prend en paramètre la file de couleurs `sequence` qui contient un certain nombre de couleurs.
   - ⋆ Le jeu électronique Simon commence par ajouter une couleur à la séquence et affiche l'intégralité de la séquence.
   - ⋆ Le joueur doit reproduire la séquence dans le même ordre ; il choisit une couleur via la fonction `saisie_joueur()`.
   - ⋆ On vérifie si cette couleur est conforme à celle de la séquence.
   - ⋆ S'il s'agit de la bonne couleur, on poursuit ; sinon, on vide `sequence`.
   - ⋆ Si le joueur arrive au bout de la séquence, il valide le tour de jeu et le jeu se poursuit avec un nouveau tour de jeu ; sinon le joueur a perdu et le jeu s'arrête.

   La fonction `tour_de_jeu` s'arrête donc si le joueur a trouvé toutes les bonnes couleurs de `sequence` dans l'ordre, ou bien dès que le joueur se trompe.
   Après l'exécution de la fonction `tour_de_jeu`, la file `sequence` ne devra pas être vidée de ses éléments en cas de victoire.

   (a) Afin d'obtenir la fonction `tour_de_jeu(sequence)` correspondant au comportement décrit ci-dessus, recopier le script ci-dessous et
   - compléter les pointillés ;
   - choisir parmi les propositions de syntaxes suivantes lesquelles correspondent aux ZONES A, B, C, D, E et F figurant dans le script et les y remplacer (il ne faut donc en choisir que six parmi les onze) :

```
vider(sequence)
defiler(sequence)
enfiler(sequence, c_joueur)
enfiler(stock, c_seq)
enfiler(sequence, defiler(stock))
enfiler(stock, defiler(sequence))
affiche_seq(sequence)
while not est_vide(sequence):
while not est_vide(stock):
if not est_vide(sequence):
if not est_vide(stock):
```

```
1  def tour_de_jeu(sequence):
2      ZONE A
3      stock = creer_file_vide()
4      while not est_vide(sequence):
5          c_joueur = saisie_joueur()
6          c_seq = ZONE B
7          if c_joueur ..... c_seq:
8              ZONE C
9          else:
10             ZONE D
11     ZONE E
12         ZONE F
```

   (b) Proposer une modification pour que la fonction se répète si le joueur trouve toutes les couleurs de la séquence (dans ce cas, une nouvelle couleur est ajoutée) ou s'il se trompe (dans ce cas, la séquence est vidée et se voit ajouter une nouvelle couleur). On pourra ajouter des instructions qui ne sont pas proposées dans la question 4.(a).