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

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2021

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

21NSIJ2AN1

Exercice 1 (4 points)

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

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

On rappelle qu'en SQL la clause `ORDER BY`, suivie d'un attribut, permet de classer les résultats par ordre croissant de l'attribut.

Dans un lycée, le parc informatique est constitué d'ordinateurs, d'imprimantes, de vidéoprojecteurs et de TNI (Tableau Numérique Interactif).
Tous les ordinateurs et toutes les imprimantes sont connectés au réseau de l'établissement.
Chaque salle de cours est identifiée par un numéro unique et contient :
- un ou plusieurs ordinateurs reliés au réseau de l'établissement ;
- aucun ou un seul vidéoprojecteur ;
- s'il y a un vidéoprojecteur, aucun ou un seul TNI ;
- une ou plusieurs imprimantes réseau.

Un ordinateur peut être connecté via le réseau à une ou plusieurs imprimantes (situées éventuellement dans une ou plusieurs autres salles) et à un vidéoprojecteur avec TNI ou non.

Les ordinateurs et les imprimantes possèdent un nom unique sur le réseau de l'établissement.

Les vidéoprojecteurs ne sont pas connectés au réseau, ils ne possèdent pas de nom unique. Ils sont donc identifiés par le numéro de la salle où ils sont installés.

Tous ces matériels sont gérés à l'aide d'une base de données relationnelle qui comprend 3 relations (ou tables) nommées : `Ordinateur`, `Videoprojecteur`, `Imprimante`.

On donne ci dessous le schéma relationnel de la relation `Ordinateur`, suivi d'un extrait de la relation `Ordinateur`. La clé primaire est soulignée.

`Ordinateur(nom_ordi : String, salle : String, marque_ordi : String, modele_ordi : String, annee : Int, video : Boolean)`

On distingue deux types d'ordinateurs :
- les ordinateurs multimédias pour les logiciels généraux avec un nom commençant par le groupe de lettres Gen.
- les ordinateurs techniques pour les logiciels qui demandent plus de ressources avec un nom commençant par le groupe de lettres Tech.

Ce groupe de lettres est suivi d'un tiret et du numéro unique de l'ordinateur pour chaque type.

Les 5 premières lignes de la relation `Ordinateur`

nom_ordi	salle	marque_ordi	modele_ordi	annee	video
Gen-24	012	HP	compaq pro 6300	2012	true
Tech-62	114	Lenovo	p300	2015	true
Gen-132	223	Dell	Inspiron Compact	2019	true
Gen-133	223	Dell	Inspiron Compact	2019	false
Gen-134	223	Dell	Inspiron Compact	2019	false

1. (a) À l'aide d'un système de gestion de base de données, on envoie au serveur la requête SQL suivante :
   `SELECT salle, marque_ordi FROM Ordinateur ;`
   Quel résultat produit cette requête sur l'extrait de la relation Ordinateur donné ci-dessus ?

   (b) Quel résultat produit la requête suivante sur l'extrait de la relation `Ordinateur` donné ci-dessus ?
   `SELECT nom_ordi, salle FROM Ordinateur WHERE video = true ;`

2. Écrire une requête SQL donnant tous les attributs des ordinateurs correspondant aux années supérieures ou égales à 2017 ordonnées par dates croissantes.

3. (a) Pour quelle raison l'attribut salle ne peut-il pas être une clé primaire pour la relation `Ordinateur` ?

(b) On donne ci-dessous un extrait de la relation `Imprimante` :

Les 5 premières lignes de la relation Imprimante

nom_imprimante	marque_imp	modele_imp	salle	nom_ordi
imp_BTS_NB	HP	Laserjet pro M15w	114	Tech-62
imp_BTS_Couleur	Canon	Megatank Pixma G5050	114	Tech-62
imp_salle-info1	Brother	2360DN	223	Gen-132
imp_salle-info1	Brother	2360DN	223	Gen-133
imp_salle-info1	Brother	2360DN	223	Gen-134

On prend `(nom_imprimante, nom_ordi)` comme clé primaire.

Écrire le schéma relationnel de la relation `Imprimante` en précisant les éventuelles clés étrangères pour les autres relations.

4. On donne ci-dessous un extrait de la relation `Videoprojecteur` :

Les 4 premières lignes de la relation `Videoprojecteur`

salle	marque_video	modele_video	tni
012	Epson	xb27	true
114	Sanyo	PLV-Z3	false
223	Optoma	HD143X	false
225	Optoma	HD143X	true

(a) Écrire une requête SQL pour ajouter à la relation `Videoprojecteur` le vidéoprojecteur nouvellement installé en salle 315 de marque NEC, modèle ME402X et non relié à un TNI.

(b) Écrire une requête SQL permettant de récupérer les attributs `salle`, `nom_ordi`, `marque_video` des ordinateurs connectés à un vidéoprojecteur équipé d'un TNI.

Exercice 2 (4 points)

Cet exercice porte sur les notions de routage, de processus et de systèmes sur puces.

Un constructeur automobile utilise des ordinateurs pour la conception de ses véhicules. Ceux-ci sont munis d'un système d'exploitation ainsi que de nombreuses applications parmi lesquelles on peut citer :

- un logiciel de traitement de texte ;
- un tableur ;
- un logiciel de Conception Assistée par Ordinateur (CAO) ;
- un système de gestion de base de donnée (SGBD).

Chaque ordinateur est équipé des périphériques classiques : clavier, souris, écran et est relié à une imprimante réseau.

1. Ce constructeur automobile intègre à ses véhicules des systèmes embarqués, comme par exemple un système de guidage par satellites (GPS), un système de freinage antiblocage (ABS) ... Ces dispositifs utilisent des systèmes sur puces (SoC : Système on a Chip). Citer deux avantages à utiliser ces systèmes sur puces plutôt qu'une architecture classique d'ordinateur.

2. Un ingénieur travaille sur son ordinateur et utilise les quatre applications citées au début de l'énoncé.

   Pendant l'exécution de ces applications, des processus mobilisent des données et sont en attente d'autres données mobilisées par d'autres processus.

   On donne ci-dessous un tableau indiquant à un instant précis l'état des processus en cours d'exécution et dans lequel D1, D2, D3, D4 et D5 sont des données.

   La lettre M signifie que la donnée est mobilisée par l'application ; la lettre A signifie que l'application est en attente de cette donnée.

   Lecture du tableau : le logiciel de traitement de texte mobilise (M) la donnée D1 et est en attente (A) de la donnée D2.

		D1	D2	D3	D4	D5
Traitement de texte		M	A	-	-	-
Tableur		A	-	-	-	M
SGBD		-	M	A	A	-
CAO		-	-	A	M	A

   Montrer que les applications s'attendent mutuellement. Comment s'appelle cette situation ?

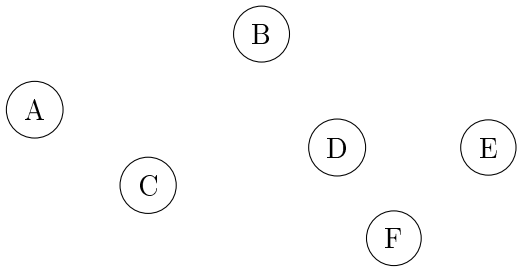
3. Ce constructeur automobile possède six sites de production qui échangent des documents entre eux. Les sites de production sont reliés entre eux par six routeurs A , B , C, D, E et F.

On donne ci dessous les tables de routage des routeurs A à F obtenus avec le protocole RIP :

Routeur A		Routeur B		Routeur C		Routeur D		Routeur E		Routeur F	
Dest	Pass	Dest	Pass	Dest	Pass	Dest	Pass	Dest	Pass	Dest	Pass
B	B	A	A	A	A	A	C	A	B	A	D
C	C	C	C	B	B	B	C	B	B	B	E
D	C	D	C	D	D	C	C	C	B	C	D
E	B	E	E	E	B	E	E	D	D	D	D
F	B	F	E	F	D	F	F	F	F	E	E

Déterminer à l'aide de ces tables le chemin emprunté par un paquet de données envoyé du routeur A vers le routeur F.

4. On veut représenter schématiquement le réseau de routeurs à partir des tables de routage. Recopier sur la copie le schéma ci-dessous :



En s'appuyant sur les tables de routage, tracer les liaisons entre les routeurs.

Exercice 3 (4 points)

Cet exercice porte sur les tableaux et sur la programmation de base en Python.

On rappelle que `len` est une fonction qui prend un tableau en paramètre et renvoie sa longueur. C'est-à-dire le nombre d'éléments présents dans le tableau.

Exemple : `len([12, 54, 34, 57])` vaut `4`.

Le but de cet exercice est de programmer différentes réductions pour un site de vente de vêtements en ligne.

On rappelle que si le prix d'un article avant réduction est de x euros,
- son prix vaut $0,5x$ si on lui applique une réduction de 50 %,
- son prix vaut $0,6x$ si on lui applique une réduction de 40 %,
- son prix vaut $0,7x$ si on lui applique une réduction de 30 %,
- son prix vaut $0,8x$ si on lui applique une réduction de 20 %,
- son prix vaut $0,9x$ si on lui applique une réduction de 10 %.

Dans le système informatique du site de vente, l'ensemble des articles qu'un client veut acheter, appelé panier, est modélisé par un tableau de flottants.

Par exemple, si un client veut acheter un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, le système informatique aura le tableau suivant :

```
tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]
```

1. (a) Écrire une fonction Python `total_hors_reduction` ayant pour argument le tableau des prix des articles du panier d'un client et renvoyant le total des prix de ces articles.

   (b) Le site de vente propose la promotion suivante comme offre de bienvenue : 20 % de réduction sur le premier article de la liste, 30 % de réduction sur le deuxième article de la liste (s'il y a au moins deux articles) et aucune réduction sur le reste des articles (s'il y en a).

   Recopier sur la copie et compléter la fonction Python `offre_bienvenue` prenant en paramètre le tableau `tab` des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre de bienvenue.

```
def offre_bienvenue(tab):
    """ tableau -> float """
    somme = 0
    longueur = len(tab)
    if longueur > 0:
        somme = tab [0] * ...
    if longueur > 1:
        somme = somme + ...
    if longueur > 2:
        for i in range(2, longueur):
            somme = ...
    return ...
```

Pour toute la suite de l'exercice, on pourra utiliser la fonction `total_hors_reduction` même si la question 1 n'a pas été traitée.

2. Lors de la période des soldes, le site de vente propose les réductions suivantes :
   - si le panier contient 5 articles ou plus, une réduction globale de 50 %,
   - si le panier contient 4 articles, une réduction globale de 40 %,
   - si le panier contient 3 articles, une réduction globale de 30 %,
   - si le panier contient 2 articles, une réduction globale de 20 %,
   - si le panier contient 1 article, une réduction globale de 10 %.

   Proposer une fonction Python `prix_solde` ayant pour argument le tableau `tab` des prix des articles du panier d'un client et renvoyant le total des prix de ces articles lorsqu'on leur applique la réduction des soldes.

3. (a) Écrire une fonction `minimum` qui prend en paramètre un tableau `tab` de nombres et renvoie la valeur minimum présente dans le tableau.

   (b) Pour ses bons clients, le site de vente propose une offre promotionnelle, à partir de 2 articles achetés, l'article le moins cher des articles commandés est offert.

   Écrire une fonction Python `offre_bon_client` ayant pour paramètre le tableau des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre bon client.

4. Afin de diminuer le stock de ses articles dans ses entrepôts, l'entreprise imagine faire l'offre suivante à ses clients : en suivant l'ordre des articles dans le panier du client, elle considère les 3 premiers articles et offre le moins cher, puis les 3 suivants et offre le moins cher et ainsi de suite jusqu'à ce qu'il reste au plus 2 articles qui n'ont alors droit à aucune réduction.

   Exemple : Si le panier du client contient un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, ce panier est représenté par le tableau suivant :

   ```
   tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]
   ```

   Pour le premier groupe (le pantalon à 30,50 euros, le tee-shirt à 15 euros, la paire de chaussettes à 6 euros), l'article le moins cher, la paire de chaussettes à 6 euros, est offert. Pour le second groupe (la jupe à 20 euros, la paire de collants à 5 euros, la robe à 35 euros), la paire de collants à 5 euros est offerte.

   Donc le total après promotion de déstockage est 111 euros.

   On constate que le prix après promotion de déstockage dépend de l'ordre dans lequel se pré- sentent les articles dans le panier.

   (a) Proposer un panier contenant les mêmes articles que ceux de l'exemple mais ayant un prix après promotion de déstockage différent de 111 euros.

   (b) Proposer un panier contenant les mêmes articles mais ayant le prix après promotion de déstockage le plus bas possible.

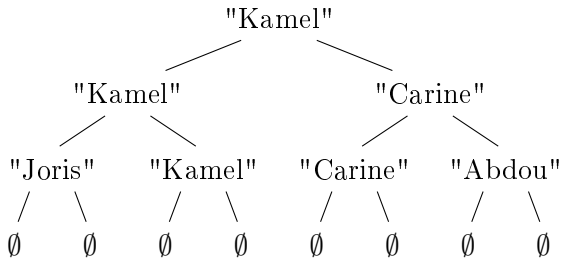
   (c) Une fois ses articles choisis, quel algorithme le client peut-il utiliser pour modifier son panier afin de s'assurer qu'il obtiendra le prix après promotion de déstockage le plus bas possible ? On ne demande pas d'écrire cet algorithme.

Exercice 4 (4 points)

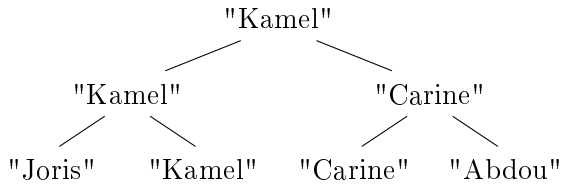
Cet exercice porte sur les arbres binaires et leurs algorithmes associés

La fédération de badminton souhaite gérer ses compétitions à l'aide d'un logiciel. Pour ce faire, une structure **arbre de compétition** a été définie récursivement de la façon suivante : un arbre de compétition est soit l'arbre vide, noté ∅, soit un triplet composé d'une chaîne de caractères appelée valeur, d'un arbre de compétition appelé sous-arbre gauche et d'un arbre de compétition appelé sous-arbre droit.

On représente graphiquement un arbre de compétition de la façon suivante :



Pour alléger la représentation d'un arbre de compétition, on ne notera pas les arbres vides, l'arbre précédent sera donc représenté par l'arbre A suivant :



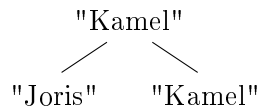
Cet arbre se lit de la façon suivante :

- 4 participants se sont affrontés : Joris, Kamel, Carine et Abdou. Leurs noms apparaissent en bas de l'arbre, ce sont les valeurs de feuilles de l'arbre.
- Au premier tour, Kamel a battu Joris et Carine a battu Abdou.
- En finale, Kamel a battu Carine, il est donc le vainqueur de la compétition.

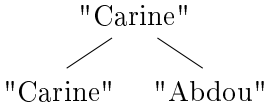
Pour s'assurer que chaque finaliste ait joué le même nombre de matchs, un arbre de compétition a toutes ses feuilles à la même hauteur. Pour toutes les questions de l'exercice, on suppose que tous les joueurs d'une même compétition ont un prénom différent.

Les quatre fonctions suivantes pourront être utilisées :

- La fonction `racine` qui prend en paramètre un arbre de compétition `arb` et renvoie la valeur de la racine.
  Exemple : en reprenant l'exemple d'arbre de compétition présenté ci-dessus, `racine(A)` vaut `"Kamel"`.
- La fonction `gauche` qui prend en paramètre un arbre de compétition `arb` et renvoie son sous-arbre gauche.
  Exemple : en reprenant l'exemple d'arbre de compétition présenté ci-dessus, `gauche(A)` vaut l'arbre représenté graphiquement ci-après :

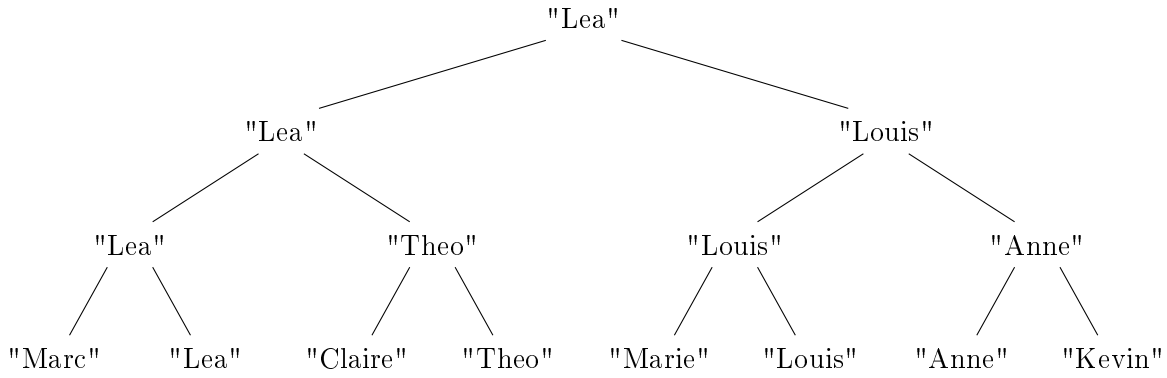


- La fonction `droit` qui prend en argument un arbre de compétition `arb` et renvoie son sous-arbre droit.
  Exemple : en reprenant l'exemple d'arbre de compétition présenté ci-dessus, `droit(A)` vaut l'arbre représenté graphiquement ci-dessous :



- La fonction `est_vide` qui prend en argument un arbre de compétition et renvoie `True` si l'arbre est vide et `False` sinon.
  Exemple : en reprenant l'exemple d'arbre de compétition présenté ci-dessus, `est_vide(A)` vaut `False`.

1. (a) On considère l'arbre de compétition `B` suivant :



Indiquer la racine de cet arbre puis donner l'ensemble des valeurs des feuilles de cet arbre.

(b) Proposer une fonction Python `vainqueur` prenant pour argument un arbre de compétition `arb` ayant au moins un joueur. Cette fonction doit renvoyer la chaîne de caractères constituée du nom du vainqueur du tournoi.

Exemple : `vainqueur(B)` vaut `"Lea"`.

(c) Proposer une fonction Python `finale` prenant pour argument un arbre de compétition `arb` ayant au moins deux joueurs. Cette fonction doit renvoyer le tableau des deux chaînes de caractères qui sont les deux compétiteurs finalistes.

Exemple : `finale(B)` vaut `["Lea", "Louis"]`.

2. (a) Proposer une fonction Python `occurrences` ayant pour paramètre un arbre de compétition `arb` et le nom d'un joueur `nom` et qui renvoie le nombre d'occurrences (d'apparitions) du joueur `nom` dans l'arbre de compétition `arb`.

Exemple : `occurences(B, "Anne")` vaut `2`.

(b) Proposer une fonction Python `a_gagne` prenant pour paramètres un arbre de compétition `arb` et le nom d'un joueur `nom` et qui renvoie le booléen `True` si le joueur nom a gagné au moins un match dans la compétition représenté par l'arbre de compétition `arb`.

Exemple : `a_gagne(B,"Louis")` vaut `True`

3. On souhaite programmer une fonction Python `nombre_matchs` qui prend pour arguments un arbre de compétition `arb` et le nom d'un joueur `nom` et qui renvoie le nombre de matchs joués par le joueur `nom` dans la compétition représentée par l'arbre de compétition `arb`.

Exemple : `nombre_matchs(B,"Lea")` doit valoir `3` et `nombre_matchs(B,"Marc")` doit valoir `1`.

(a) Expliquer pourquoi les instructions suivantes renvoient une valeur erronée. On pourra pour cela identifier le nœud de l'arbre qui provoque une erreur.

```
def nombre_matchs(arb,nom):
    """ arbre_competition , str -> int """
    return occurrences(arb,nom)
```

(b) Proposer une correction pour la fonction `nombre_matchs`.

4. Recopier et compléter la fonction `liste_joueurs` qui prend pour argument un arbre de compétition `arb` et qui renvoie un tableau contenant les participants au tournoi, chaque nom ne devant figurer qu'une seule fois dans le tableau. L'opération `+` à la ligne 8 permet de concaténer deux tableaux.

Exemple : Si `L1 = [4, 6, 2]` et `L2 = [3, 5, 1]`, l'instruction `L1 + L2` va renvoyer le tableau `[4, 6, 2, 3, 5, 1]`

```
def liste_joueurs(arb):
    """ arbre_competition -> tableau """
    if est_vide ( arb ):
        return ...
    elif ... and ... :
        return [racine(arb)]
    else :
        return ...+ liste_joueurs(droit(arb))
```

Exercice 5 (4 points)

Cet exercice porte sur la notion de pile, de file et sur la programmation de base en Python.

Les interfaces des structures de données abstraites `Pile` et `File` sont proposées ci-dessous.
On utilisera uniquement les fonctions ci-dessous :

Structure de données abstraite : `Pile`
Utilise : Élément, Booléen
Opérations : • `creer_pile_vide` : $\emptyset \to$ `Pile` `creer_pile_vide()` renvoie une pile vide • `est_vide` : `Pile` $\to$ Booléen `est_vide(pile)` renvoie `True` si `pile` est vide, `False` sinon • `empiler` : `Pile`, `Élément` $\to \emptyset$ `empiler(pile, element)` ajoute `element` à la pile `pile` • `depiler` : Pile $\to$ Élément `depiler(pile)` renvoie l'élément au sommet de la pile `pile` en le retirant de la pile

Structure de données abstraite : `File`
Utilise : Élément, Booléen
Opérations : • `creer_file_vide` : $\emptyset \to$ `File` `creer_file_vide()` renvoie une file vide • `est_vide` : `File` $\to$ Booléen `est_vide(file)` renvoie `True` si `file` est vide, `False` sinon • `enfiler` : `File`, `Élément` $\to \emptyset$ `enfiler(file, element)` ajoute `element` à la file `file` • `defiler` : File $\to$ Élément `defiler(file)` renvoie l'élément au début de la file `file` en le retirant de la file

1. (a) On considère la file F suivante :

enfilement ⟶ "rouge" "vert" "jaune" "rouge" "jaune" ⟶ défilement

Quel sera le contenu de la pile P et de la file F après l'exécution du programme Python suivant ?

```
P = creer_pile_vide ()
while not (est_vide(F)):
    empiler (P, defiler(F))
```

(b) Créer une fonction `taille_file` qui prend en paramètre une file `F` et qui renvoie le nombre d'éléments qu'elle contient. Après appel de cette fonction la file `F` doit avoir retrouvé son état d'origine.

```
def taille_file ( F ):
    """ File -> Int """
```

2. Écrire une fonction `former_pile` qui prend en paramètre une file `F` et qui renvoie une pile `P` contenant les mêmes éléments que la file. Le premier élément sorti de la file devra se trouver au sommet de la pile ; le deuxième élément sorti de la file devra se trouver juste en-dessous du sommet, etc.

Exemple : si `F` = "rouge" "vert" "jaune" "rouge" "jaune"
alors l'appel `former_pile(F)` va renvoyer la pile `P` ci-dessous :

P=

"jaune"
"rouge"
"jaune"
"vert"
"rouge"

3. Écrire une fonction `nb_elements` qui prend en paramètres une file `F` et un élément `elt` et qui renvoie le nombre de fois où `elt` est présent dans la file `F`.
   Après appel de cette fonction la file `F` doit avoir retrouvé son état d'origine.
4. Écrire une fonction `verifier_contenu` qui prend en paramètres une file `F` et trois entiers : `nb_rouge`, `nb_vert` et `nb_jaune`.
   Cette fonction renvoie le booléen `True` si "rouge" apparaît au plus `nb_rouge` fois dans la file `F`, "vert" apparaît au plus `nb_vert` fois dans la file `F` et "jaune" apparaît au plus `nb_jaune` fois dans la file `F`. Elle renvoie `False` sinon. On pourra utiliser les fonctions précédentes.