BACCALAURÉAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES
Partie écrite
Classe terminale de la voie générale

Bac Blanc

DURÉE DE L'ÉPREUVE : 3 heures 30 min

Le sujet comporte 11 pages numérotées de 1/11 à 11/11.
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit choisir 3 exercices qu'il traitera sur les 5 exercices proposés.
Il est rappelé que la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies.

Exercice 1 (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` : $\emptyset \longrightarrow$ `Pile`
`creer_pile_vide()` renvoie une pile vide
• `est_vide` : `Pile` $\longrightarrow$ `Booléen`
`est_vide(pile)` renvoie True si `pile` est vide, False sinon
• `empiler` : `Pile, Élément` $\longrightarrow \emptyset$
`empiler(pile, element)` ajoute `element` à la pile `pile`
• `depiler` : `Pile` $\longrightarrow$ `Élément`
`depiler(pile)` renvoie l'élément au sommet de la `pile` en le retirant de la `pile`

Structure de données abstraite : File
Utilise : Élément, Booléen

Opérations :
• `creer_file_vide` : $\emptyset \longrightarrow$ `File`
`creer_file_vide()` renvoie une file vide
• `est_vide` : `File` $\longrightarrow$ `Booléen`
`est_vide(file)` renvoie True si `file` est vide, False sinon
• `enfiler` : `File, Élément` $\longrightarrow \emptyset$
`enfiler(file, element)` ajoute `element` dans la file `file`
• `defiler` : `File` $\longrightarrow$ `Élément`
`defiler(file)` renvoie l'élément au sommet de la file `file` en le retirant de la file `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 ?

```
1 P = creer_pile_vide()
2 while not(est_vide(F)):
3     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):
    """F -> 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.

EXERCICE 2 (4 points)

Cet exercice porte sur la programmation en général et la récursivité en particulier.

On s'intéresse dans cet exercice à un algorithme de mélange des éléments d'une liste.

1. Pour la suite, il sera utile de disposer d'une fonction `echange` qui permet d'échanger dans une liste `lst` les éléments d'indice `i1` et `i2`.
Expliquer pourquoi le code Python ci-dessous ne réalise pas cet échange et en proposer une modification.

```
def echange(lst, i1, i2):
    lst[i2] = lst[i1]
    lst[i1] = lst[i2]
```

2. La documentation du module `random` de Python fournit les informations ci-dessous concernant la fonction `randint(a,b)` :

```
Renvoie un entier aléatoire N tel que a <= N <= b. Alias
pour randrange(a,b+1).
```

Parmi les valeurs ci-dessous, quelles sont celles qui peuvent être renvoyées par l'appel `randint(0, 10)` ?

`0` `1` `3.5` `9` `10` `11`

3. Le mélange de Fischer Yates est un algorithme permettant de permuter aléatoirement les éléments d'une liste. On donne ci-dessous une mise en œuvre récursive de cet algorithme en Python.

```
from random import randint

def melange(lst, ind):
    print(lst)
    if ind > 0:
        j = randint(0, ind)
        echange(lst, ind, j)
        melange(lst, ind-1)
```

a. Expliquer pourquoi la fonction `melange` se termine toujours.

b. Lors de l'appel de la fonction `melange`, la valeur du paramètre `ind` doit être égal au plus grand indice possible de la liste `lst`.
Pour une liste de longueur n, quel est le nombre d'appels récursifs de la fonction `melange` effectués, sans compter l'appel initial ?

c. On considère le script ci-dessous :

```
lst = [v for v in range(5)]
melange(lst, 4)
```

On suppose que les valeurs successivement renvoyées par la fonction `randint` sont `2, 1, 2` et `0`.

Les deux premiers affichages produits par l'instruction `print(lst)` de la fonction `melange` sont :

```
[0, 1, 2, 3, 4]
[0, 1, 4, 3, 2]
```

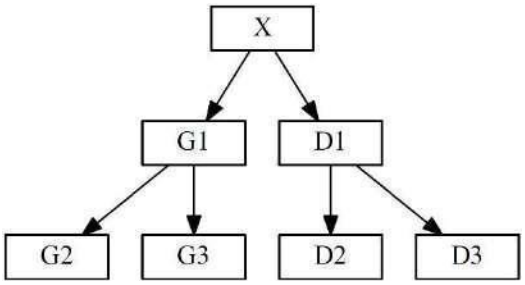
Donner les affichages suivants produits par la fonction `melange`.

d. Proposer une version itérative du mélange de Fischer Yates.

Exercice 3 (4 points)

Notion abordée : les arbres binaires de recherche.

Un arbre binaire est soit vide, soit un nœud qui a une valeur et au plus deux fils (le sous-arbre gauche et le sous-arbre droit).



X est un nœud, sa valeur est X.valeur

G1 est le fils gauche de X, noté X.fils_gauche

D1 est le fils droit de X, noté X.fils_droit

Un arbre binaire de recherche est ordonné de la manière suivante :

Pour **chaque** nœud X,

- les valeurs de tous les nœuds du sous-arbre gauche sont **strictement inférieures** à la valeur du nœud X
- les valeurs de tous les nœuds du sous-arbre droit sont **supérieures ou égales** à la valeur du nœud X

Ainsi, par exemple, toutes les valeurs des nœuds G1, G2 et G3 sont strictement inférieures à la valeur du nœud X et toutes les valeurs des nœuds D1, D2 et D3 sont supérieures ou égales à la valeur du nœud X.

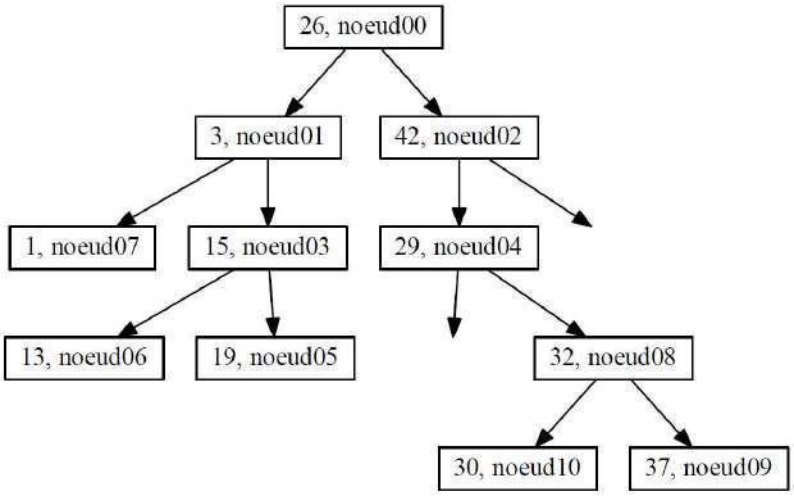
Voici un exemple d'arbre binaire de recherche dans lequel on a stocké dans cet ordre les valeurs :

[26, 3, 42, 15, 29, 19, 13, 1, 32, 37, 30]

L'étiquette d'un nœud indique la valeur du nœud suivie du nom du nœud.

Les nœuds ont été nommés dans l'ordre de leur insertion dans l'arbre ci-dessous.

`'29, noeud04'` signifie que le nœud nommé `noeud04` possède la valeur `29`.



1. On insère la valeur 25 dans l'arbre, dans un nouveau nœud nommé nœud11.
   Recopier l'arbre binaire de recherche étudié et placer la valeur 25 sur cet arbre en coloriant en rouge le chemin parcouru.
   Préciser sous quel nœud la valeur 25 sera insérée et si elle est insérée en fils gauche ou en fils droit, et expliquer toutes les étapes de la décision.

2. **Préciser** toutes les valeurs entières que l'on peut stocker dans le nœud fils gauche du nœud04 (vide pour l'instant), en respectant les règles sur les arbres binaires de recherche ?

3. Voici un algorithme récursif permettant de parcourir et d'afficher les valeurs de l'arbre :

```
Parcours(A)        # A est un arbre binaire de recherche
  Afficher(A.valeur)
  Parcours(A.fils_gauche)
  Parcours(A.fils_droit)
```

   3.a. **Écrire** la liste de toutes les valeurs dans l'ordre où elles seront affichées.
   3.b. **Choisir** le type de parcours d'arbres binaires de recherche réalisé parmi les propositions suivantes : Préfixe, Suffixe ou Infixe

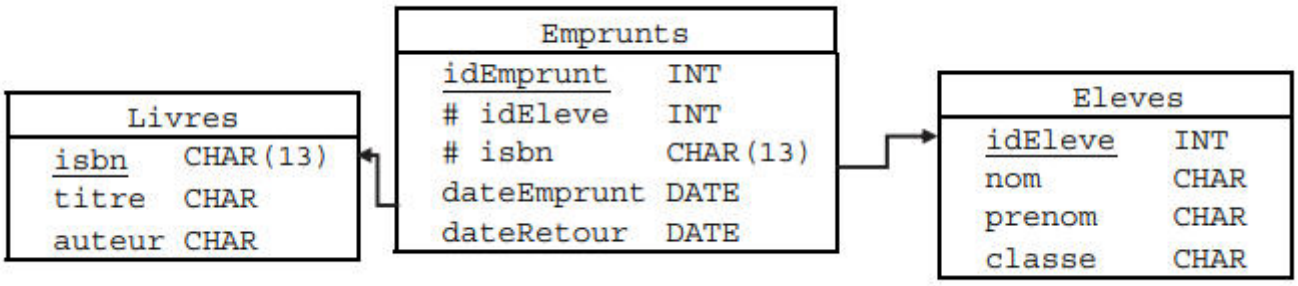
4. En vous inspirant de l'algorithme précédent, écrire un algorithme `Parcours2` permettant de parcourir et d'afficher les valeurs de l'arbre `A` dans l'ordre croissant.

EXERCICE 4 (4 points)

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

L'énoncé de cet exercice utilise les mots du langage SQL suivants :
`SELECT FROM, WHERE, JOIN ON, INSERT INTO VALUES, UPDATE, SET, DELETE, COUNT, AND, OR.`

On considère dans cet exercice une gestion simplifiée des emprunts des ouvrages d'un CDI. La base de données utilisée sera constituée de trois relations (ou tables) nommées `Eleves`, `Livres` et `Emprunts` selon le schéma relationnel suivant :



Dans ce schéma relationnel, un attribut souligné indique qu'il s'agit d'une clé primaire. Le symbole # devant un attribut indique qu'il s'agit d'une clé étrangère et la flèche associée indique l'attribut référencé. Ainsi, l'attribut `idEleve` de la relation `Emprunts` est une clé étrangère qui fait référence à la clé primaire `idEleve` de la relation `Eleves`.

1. Expliquer pourquoi le code SQL ci-dessous provoque une erreur.

```
INSERT INTO Eleves VALUES (128, 'Dupont', 'Jean',  'T1') ;
INSERT INTO Eleves VALUES (200, 'Dupont', 'Jean',  'T1') ;
INSERT INTO Eleves VALUES (128, 'Dubois', 'Jean',  'T2') ;
```

2. Dans la définition de la relation `Emprunts`, qu'est-ce qui assure qu'on ne peut pas enregistrer un emprunt pour un élève qui n'a pas encore été inscrit dans la relation `Eleves` ?

3. Écrire une requête SQL qui renvoie les titres des ouvrages de Molière détenus par le CDI.

4. Décrire le résultat renvoyé par la requête ci-dessous.

```
SELECT COUNT(*)
FROM Eleves
WHERE classe = 'T2' ;
```

5. Camille a emprunté le livre *Les misérables*. Le code ci-dessous a permis d'enregistrer cet emprunt.

```
INSERT INTO Emprunts
VALUES  (640,  192,  '9782070409228',  '2020-09-15',  NULL);
```

Camille a restitué le livre le 30 septembre 2020.
Recopier et compléter la requête ci-dessous de manière à mettre à jour la date de retour dans la base de données.

```
UPDATE Emprunts SET …… WHERE …… ;
```

6. Décrire le résultat renvoyé par la requête ci-dessous.

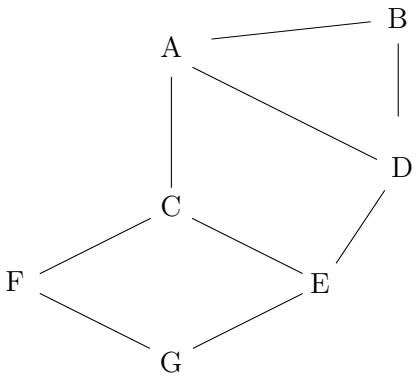
```
SELECT DISTINCT nom, prenom
FROM Eleves, Emprunts
WHERE Eleves.idEleve = Emprunts.idEleve
AND Eleves.classe = 'T2' ;
```

7. Écrire une requête SQL qui permet de lister les noms et prénoms des élèves qui ont emprunté le livre *Les misérables*.

Exercice 5 (4 points)

Cet exercice porte sur les réseaux en général et les protocoles RIP et OSPF en particulier.

On considère un réseau composé de plusieurs routeurs reliés de la façon suivante :



Le protocole RIP

Le protocole RIP permet de construire les tables de routage des différents routeurs, en indiquant pour chaque routeur la distance, en nombre de sauts, qui le sépare d'un autre routeur.
Pour le réseau ci-dessus, on dispose des tables de routage suivantes :

Table de routage du routeur A		
Destination	Routeur suivant	Distance
B	B	1
C	C	1
D	D	1
E	C	2
F	C	2
G	C	3

Table de routage du routeur B		
Destination	Routeur suivant	Distance
A	A	1
C	A	2
D	D	1
E	D	2
F	A	3
G	D	3

Table de routage du routeur C		
Destination	Routeur suivant	Distance
A	A	1
B	A	2
D	E	2
E	E	1
F	F	1
G	F	2

Table de routage du routeur D		
Destination	Routeur suivant	Distance
A	A	1
B	B	1
C	E	2
E	E	1
F	A	3
G	E	2

Table de routage du routeur E		
Destination	Routeur suivant	Distance
A	C	2
B	D	2
C	C	1
D	D	1
F	G	2
G	G	1

Table de routage du routeur F		
Destination	Routeur suivant	Distance
A	C	2
B	C	3
C	C	1
D	C	3
E	G	2
G	G	1

Question 1

1. Le routeur A doit transmettre un message au routeur G, en effectuant un nombre minimal de sauts. Déterminer le trajet parcouru.
2. Déterminer une table de routage possible pour le routeur G obtenu à l'aide du protocole RIP.

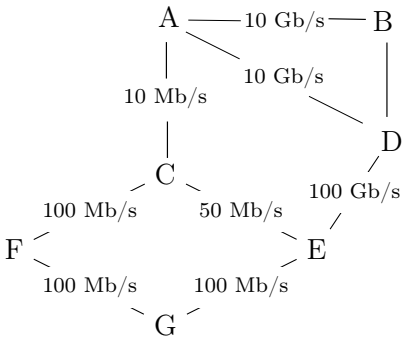
Question 2 Le routeur C tombe en panne. Reconstruire la table de routage du routeur A en suivant le protocole RIP.

Le protocole OSPF

Contrairement au protocole RIP, l'objectif n'est plus de minimiser le nombre de routeurs traversés par un paquet. La notion de distance utilisée dans le protocole OSPF est uniquement liée aux coûts des liaisons. L'objectif est alors de minimiser la somme des coûts des liaisons traversées.
Le coût d'une liaison est donné par la formule suivante :

$$\text{coût} = \frac{10^8}{d}$$

où d est la bande passante en bits/s entre les deux routeurs.
On a rajouté sur le graphe représentant le réseau précédent les différents débits des liaisons.
On rappelle que 1 Gb/s = 1 000 Mb/s = 10^9 bits/s.



Question 3

1. Vérifier que le coût de la liaison entre les routeurs A et B est 0,01.
2. La liaison entre le routeur B et D a un coût de 5. Quel est le débit de cette liaison ?

Question 4

Le routeur A doit transmettre un message au routeur G, en empruntant le chemin dont la somme des coûts sera la plus petite possible. Déterminer le chemin parcouru. On indiquera le raisonnement utilisé.