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

Jour 1

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 13 pages numérotées de 1/13 à 13/13.

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

Chaque exercice est noté sur 4 points.

EXERCICE 1 (4 points)

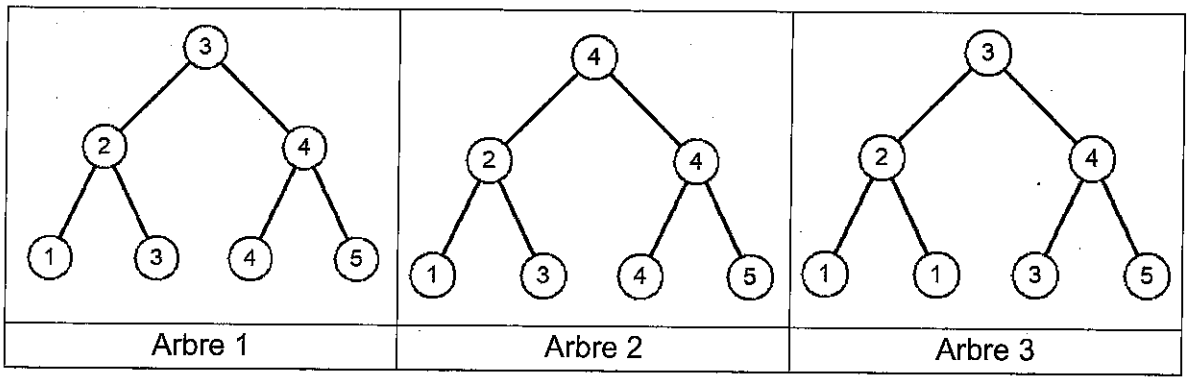
Cet exercice porte sur le thème "Algorithmique", les arbres binaires de recherche et leurs parcours.

Rappel : Un arbre binaire de recherche (ABR) est un arbre binaire étiqueté avec des clés tel que :

- Les clés du sous arbre gauche sont inférieures ou égales à celle de la racine ;
- Les clés du sous arbre droit sont strictement supérieures à celle de la racine ;
- Les deux sous arbres sont eux-mêmes des arbres binaires de recherche.

Partie A : Préambule

1) Recopier sur votre copie le ou les numéro(s) correspondant aux arbres binaires de recherche parmi les arbres suivants :



Arbre 1	Arbre 2	Arbre 3

Partie B : Analyse

On considère la structure de données abstraites ABR (Arbre Binaire de Recherche) que l'on munit des opérations suivantes :

```
Structure de données : ABR
Utilise : Booleen, Element
Opérations :
```

- `creer_arbre : ∅ →ABR`

  `creer_arbre()` renvoie un arbre vide.

- `est_vide : ABR→Booleen`

  `est_vide(a)` renvoie True si l'arbre a est vide et False sinon.

- `racine : ABR→ Element`

  `racine(a)` renvoie la clé de la racine de l'arbre non vide a.

- `sous_arbre_gauche : ABR→ABR`

  `sous_arbre_gauche(a)` renvoie le sous-arbre gauche de l'arbre non vide a.

- `sous_arbre_droit : ABR→ABR`

  `sous_arbre_droit(a)` renvoie le sous-arbre droit de l'arbre non vide a.

- `inserer : ABR, Element→Rien`
  `inserer(a, e)` insère la clé e dans l'arbre a.

2.a) Dans un ABR, où se trouve le plus petit élément ? Justifier.

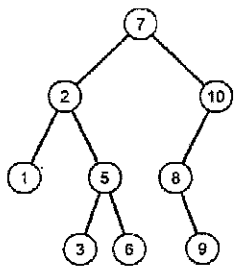
Pour rechercher une clé dans un ABR, il faut comparer la clé donnée avec la clé située à la racine. Si cette clé est à la racine, la fonction renvoie vrai sinon il faut procéder récursivement sur les sous arbres à gauche ou à droite.

2.b) En utilisant les fonctions ci-dessus, écrire une fonction récursive **`RechercheValeur`** prenant en arguments la clé recherchée et l'arbre ABR considéré. Cette fonction retourne un booléen (vrai ou faux) indiquant si la clé est présente dans l'arbre ou non.

3. On considère l'ABR ci-contre :

3.a) Dire à quel type de parcours correspond le résultat suivant où les clés sont triées dans l'ordre croissant :

1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 - 10



3.b) Donner le parcours préfixe de cet arbre.

3.c) Donner le parcours suffixe de cet arbre.

3.d) Donner le parcours en largeur de cet arbre.

EXERCICE 2 (4 points)

Cet exercice aborde les notions de classes, itération et récursivité

Une petite société immobilière ne voulant pas investir dans une base de données nécessitant une mise en place longue et fastidieuse a créé un fichier .csv pour stocker ses annonces. La consultation et la mise à jour de ce fichier ne seront pas étudiées ici. Pour limiter notre étude, nous considérerons que les données sont stockées temporairement dans une liste v dont voici la structure simplifiée :

```
# pieces
class Piece:
    def __init__(self, a, b):
        self.nom = a   # nom
        self.sup = b   #superficie de la piece
    def sup(self):
        return self.sup
#villas
class Villa:
    def __init__(self, a, b, c, d, e):
        self.nom = a # nom de la villa
        self.sejour = b  # caracteristiques sejour
        self.ch1   = c # caracteristiques de la 1ere chambre
        self.ch2   = d # caracteristiques de la 2eme chambre
        self.eqCuis = e  # equipement de la cuisine "eq" ou "noneq"
    def nom(self):
        return self.nom
    def surface(self):
        return  … …
    def equip(self):
        return self.eqCuis

# Programme principal
v=[]
v.append( Villa("Les quatre vents", Piece("séjour",40),
Piece("ch1",10), Piece("ch2",20),"eq"))
v.append( Villa("Les goélands", Piece("séjour",50),
Piece("ch1",15), Piece("ch2",15), "eq"))
v.append( Villa("Rêve d'été", Piece("séjour",30), Piece("ch1",15),
Piece("ch2",20), "non eq"))
v.append( Villa("Les oliviers", Piece("séjour",30),
Piece("ch1",10), Piece("ch2",20), "eq"))
v.append( Villa("Bellevue", Piece("séjour",30), Piece("ch1",10),
Piece("ch2",20), "non eq"))
```

La structure de données retenue pour l'exercice est définie par la classe Villa.

Partie A : Analyse du code et complétion

1.a) Combien d'éléments contient la liste `v` ?

1.b) Que retourne l'instruction `v[1].nom()` ?

Pour accéder à l'information de la surface habitable du logement, le développeur souhaite ajouter la méthode `surface()` qui renvoie cette surface.

1.c) Compléter sur votre copie la méthode `surface()`.

```
def surface(self):
    return  ... ...
```

L'agent immobilier veut pouvoir consulter les villas qui disposent d'une cuisine équipée. Pour cela vous devez écrire la portion du programme qui affichera la liste des villas équipées. Elle devra parcourir séquentiellement la liste et afficher à l'écran le nom de chaque villa équipée.

2. Rédigez sur votre copie la portion de programme réalisant cette sélection.

Partie B : Récursivité

L'agent immobilier veut répondre à une demande d'un client : " Quelle est votre villa la plus grande ?" Pour cela, il souhaite disposer d'une fonction `max_surface()` qui va extraire de la liste `v`, la villa désirée. Nous avons décidé d'écrire cette fonction de façon récursive.

3. Recopiez parmi les propositions suivantes celle qui caractérise un appel récursif.

- appel d'une fonction par elle-même.
- appel dont l'exécution est un processus itératif.
- appel d'une fonction comportant une boucle.

L'algorithme suivant a été choisi :

Il faut partir d'une liste de villas :

- si cette liste contient un seul élément, c'est le résultat ;
- si la liste en contient plusieurs, il faut analyser les deux premiers éléments, éliminer la villa de plus petite surface, et recommencer avec la liste tronquée.

A la fin du processus, une seule villa est renvoyée.

Pour écrire cette fonction, dans un premier temps, nous allons donc distinguer deux cas:

- celui où la liste des villas ne contient qu'une villa :

    il faut renvoyer la villa

- celui où la liste en contient au moins deux :

    si la surface de la villa `v[0]` est inférieure à celle de la villa `v[1]`

        il faut supprimer `v[0]` de la liste

    sinon

        il faut supprimer `v[1]` de la liste.

4. Ecrivez sur votre copie le code de cette fonction en Python.

```
def max_surface(v):
....
```

EXERCICE 3 (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 suivant :

`DELETE, FROM, INSERT INTO, JOIN, SELECT, SET, UPDATE, VALUES, WHERE.`

Les clés primaires seront soulignées et les clés étrangères seront précédées d'un #.

Le satellite GAIA a pour mission de cartographier un très grand nombre d'objets autour du Système Solaire. Régulièrement un catalogue est produit pour publier les données obtenues. Il est disponible sous différents formats dont par exemple sous forme de fichier csv. `Num_Objet` identifie chaque objet cartographié de manière unique.

Voici un extrait du catalogue :

Num_Objet	Num_Systeme	Nom_Systeme	Type_Objet	Nom_Objet	Ascension_Droite	Declinaison	Parallaxe	Nom_SIMBAD
1	1	alf Cen	LM	Proxima Cen	217,392	-62,676	768,067	alf Cen C
2	1	alf Cen	Planet	Proxima Cen b	217,392	-62,676	768,067	
3	1	alf Cen	*	alf Cen A	219,902	-60,834	743,000	alf Cen A
4	1	alf Cen	*	alf Cen B	219,896	-60,838	743,000	alf Cen B
5	2	Barnard's Star	LM	Barnard's Star	269,449	4,739	546,976	Barnard's Star
6	3	Luhman 16	BD	Luhman 16 A	162,309	-53,318	501,557	Luhman 16A
7	3	Luhman 16	BD	Luhman 16 B	162,308	-53,318	501,557	Luhman 16B
9	5	Wolf 359	LM	Wolf 359	164,103	7,003	415,179	Wolf 359
10	6	HD 95735	LM	HD 95735	165,831	35,949	392,753	HD 95735
11	6	HD 95735	Planet	Lalande 21185 b	165,831	35,949	392,753	
12	7	alf CMa	*	alf CMa A	101,287	-16,716	379,210	alf CMa A
13	7	alf CMa	WD	alf CMa B	101,287	-16,721	374,490	alf CMa B
14	8	G 272-61	LM	G 272-61 A	24,772	-17,948	367,712	G 272-61A
15	8	G 272-61	LM	G 272-61 B	24,772	-17,948	373,844	G 272-61B
16	9	V1216 Sgr	LM	Ross 154	282,459	-23,837	336,027	Ross 154
17	10	HH And	LM	Ross 248	355,480	44,170	316,481	Ross 248

Pour manipuler plus facilement les données, un chercheur utilise un système de base de données relationnelle, dans lequel il crée le schéma relationnel de la table Gaia :

`Gaia (Num_Objet : Int, Num_Systeme : Int, Nom_Systeme : String, #Type_Objet : String, Nom_Objet : String, Ascension_Droite : Real, Declinaison : Real, Parallaxe : Real, Nom_SIMBAD : String)`

Partie A : Schéma relationnel

1. Justifier que l'attribut `Num_Objet` peut être choisi comme clé primaire de la table Gaia.

Le type de l'objet (`Type_Objet`) n'est pas une information directement compréhensible et le chercheur décide de créer une nouvelle table appelée Type. La clé primaire est `Type_Objet`

Soit la table `Type` contenant les informations suivantes :

Type_Objet	Libelle_Objet
LM	Etoile de faible masse
Planet	Planète
*	Etoile
BD	Naine Brune
WD	Naine Blanche

2. Proposer le schéma relationnel de la table `Type` en soulignant la clé primaire.

Partie B : Instructions SQL

3. Parmi les commandes suivantes, laquelle ne provoque pas d'erreur ?

a)
```
INSERT INTO Gaia VALUES ('8', 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', 133.781,-7.244, 439.000, 'WISEA J085510');
```
b)
```
INSERT INTO Gaia VALUES (8, 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', 133.781,-7.244, 439.000, 'WISEA J085510');
```
c)
```
INSERT INTO Gaia VALUES (8, 4, WISEA J085510, 'Naine Brune',
 WISEA J085510, 133.781,-7.244, 439.000, WISEA J085510);
```
d)
```
INSERT INTO Gaia VALUES (8, 4, 'WISEA J085510', 'Naine Brune',
'WISEA J085510', '133.781',-7.244, 439.000, 'WISEA J085510')
```

4. Expliquer pourquoi le code SQL suivant ne fonctionne pas :

```
INSERT INTO Type VALUES ('BD', 'Trou Noir');
```

5. Indiquer le résultat de la requête suivante exécutée sur l'extrait présenté :

```
SELECT Nom_Objet, Parallaxe FROM Gaia WHERE Type_Objet = 'Planet';
```

6. Ecrire la requête qui permet de récupérer le nom du système, le nom de l'objet et le libellé du type pour des objets ayant une parallaxe supérieure à 400 millisecondes d'arc et étant des 'Etoile'.

On veut remplacer la valeur '*' de `Type_Objet` dans les tables Gaia et Type, par la valeur 'ST'.

7.a) Ecrire une requête permettant d'insérer un nouveau type 'ST' de libéllé 'Etoile'.

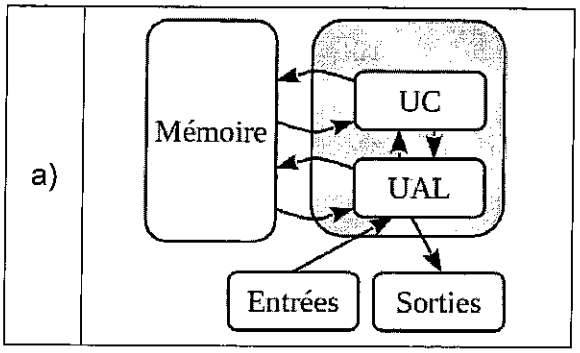
7.b) Expliquer la démarche à suivre et les commandes à exécuter pour finaliser la modification.

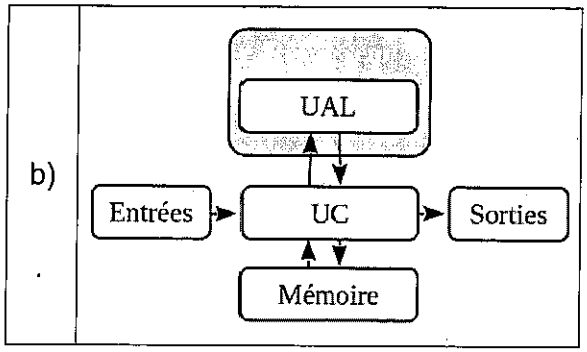
EXERCICE 4 (4 points)

Cet exercice porte sur le thème "Architecture matérielle, gestion de processus et réseaux".

Partie A : Architecture matérielle

1. Parmi les schémas suivants, lequel représente le mieux une architecture de Von Neumann ?



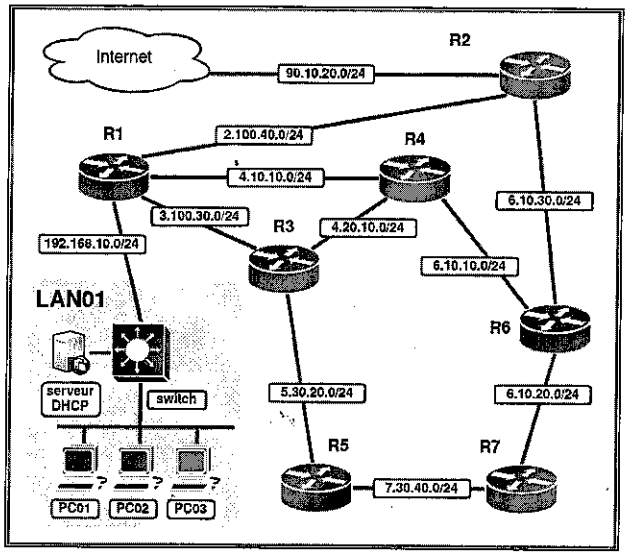


Partie B : Le réseau

Rappelons qu'une adresse IPv4 est composée de 4 octets. Elle est notée `W.X.Y.Z`, où W, X, Y et Z sont les valeurs des 4 octets, exprimées en décimal.

La notation CIDR `W.X.Y.Z/n` désigne une adresse IPv4 et signifie que les **n** premiers bits de poids forts de cette adresse représentent la partie « réseau », les bits suivants de poids faibles représentent la partie « machine ». Toutes les adresses des machines connectées à un réseau local ont la même partie réseau.

On considère le schéma de réseau suivant :



2. Proposer une adresse IPv4 pour l'ordinateur PC02.

3. Compte-tenu du masque de sous-réseau de LAN01, combien de machines peuvent être connectées ?

4. Quel est le rôle d'un switch ?

5. Quel est le rôle d'un routeur ?

Ci-dessous, une représentation partielle de la table de routage RIP du routeur R1 :

Table de routage de R1

Destination Réseau	Passerelle	Métrique
192.168.10.0/24	0.0.0.0	0
2.100.40.0/24	2.100.40.1	1
3.100.30.0/24	3.100.30.2	1
4.10.10.0/24	4.10.10.2	1
4.20.10.0/24	...	...
7.30.40.0/24	...	...
6.10.30.0/24	...	...
90.10.20.0/24	2.100.40.1	2

6. Reproduire et compléter sur votre copie les lignes incomplètes de cette table en suivant la logique du protocole RIP.

Suite à un problème technique la liaison de R1 vers R2 n'est plus active.

7. Écrire sur votre copie la ligne correspondant à la destination « Internet », telle qu'elle serait modifiée selon le protocole RIP.

EXERCICE 5 (4 points)

Cet exercice porte sur la notion de file et sur la programmation en Python du programme de Terminale.

Rappel : une file est une structure de données abstraite fondée sur le principe « premier arrivé, premier servi. »

1. Laquelle de ces deux situations est associée à une structure de file ?

Situation 1 : « *Je cuisine des crêpes. Dès qu'une crêpe est faite, je la place sur un plat. Chaque nouvelle crêpe est placée sur la crêpe précédente. Quand je vais manger une de ces crêpes, je commencerai par la crêpe située en haut de mon tas.* »

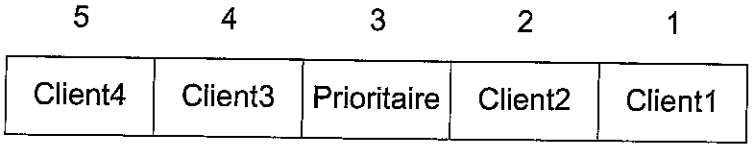
Situation 2 : « *Je dispose d'une imprimante placée en réseau dans ma salle de classe équipée d'ordinateurs, tous en réseau. Tous les élèves présents ont accès à cette imprimante, via le réseau. A la fin de la séance, les élèves envoient leur production à l'impression. Les documents sont imprimés dans l'ordre d'arrivée* ».

On modélise la gestion de l'attente à une caisse de supermarché. Les clients sont associés à une **File**. Les personnes prioritaires passeront devant les autres clients sans attendre. Nous ne tenons pas compte dans cet exercice de graduation dans les « priorités ». Nous ne tenons pas compte de personnes arrivant ensemble en caisse : il y aura toujours un des deux clients arrivé avant l'autre. On appelle 1ère personne dans la queue, la première personne qui est juste derrière le client en train de payer ses articles en caisse. En d'autres termes, le client qui règle ses articles ne compte plus, puisqu'il n'attend plus dans la queue.

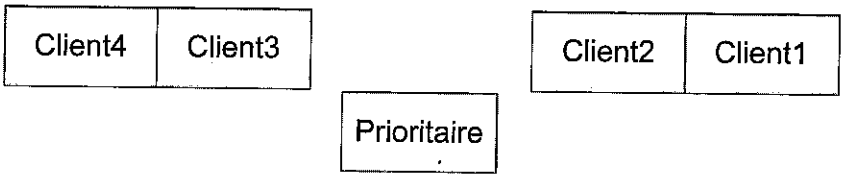
Voici les règles appliquées :

- La 1ère personne arrivée se place dans la queue.
- Le contrôleur « relations clients » du supermarché vérifie les priorités « clients ».
- Si une personne dispose d'un accès prioritaire, elle passe en position 1, et de ce fait, tout le reste des clients dans l'attente rétrograde d'une place.
- Si deux personnes sont prioritaires, la 2ème arrivée se placera derrière la 1ère arrivée « Prioritaire » et ainsi de suite avec tout nouveau prioritaire.

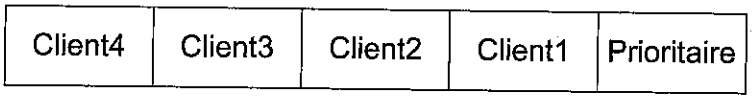
Exemple : À un instant t, la file est dans l'état ci-dessous :



La réorganisation grâce au contrôleur « relations clients » se met en place : « Client1 » et « Client2 » font un pas de côté, de même pour les personnes derrière « Prioritaire », en respectant leur ordre d'arrivée.



Le « Prioritaire » s'avance et se retrouve en position 1. Puis la file finale se réorganise.



Nous utiliserons uniquement les quatre fonctions primitives suivantes pour la suite des questions.

Structure de données abstraite: **File**

Utilise: Élément, Booléen

Opérations :

- creer_file_vide : $\emptyset \rightarrow$ File

  creer_file_vide() renvoie une file vide

- est_vide : File → Booléen

  est_vide (File) renvoie True si File est vide, False sinon

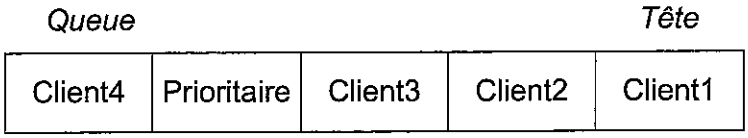
- enfiler: File, Élément → Rien

  enfiler(File, element) ajoute element dans la file File

- defiler: File → Élément

  defiler(File) renvoie l'élément en tête de la file File tout en le retirant de la file

On suppose que le contenu de la file `F` est le suivant:



2.a) On considère la file `V` définie dans le code ci-dessous.

Quel sera le contenu de `V`, `F` et de la variable val à la suite de ces instructions Python?

```
V = creer_file_vide()
val = defiler(F)
while  not est_vide (F) and val != 'Prioritaire'
      enfiler (V, val)
      val = defiler(F)
```

On considère la fonction **`longueur_file`**, écrite en Python, ci-dessous. Le but de cette fonction est de renvoyer le nombre d'éléments d'une file donnée en paramètre. À la fin du programme, la file `F` doit avoir retrouvé son état d'origine.

2.b) Compléter le programme ci-dessous.

```
def longueur_file(F) :
      V= creer_file_vide()
      n= 0
      while not est_vide(F) :
         n = …
         val = defiler(F)
         enfiler(V,val)
      while not est_vide(V) :
         …
         …
      return n
```

2.c) Écrire une fonction **`compter_prio`** qui prend en paramètre une file `F` . Cette fonction renvoie le nombre de personnes prioritaires dans la file d'attente, à l'instant t. La file `F` doit être identique à celle du départ en fin d'éxécution de la fonction.