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 2

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 la notion de file et sur la programmation de base en Python.

On rappelle qu'une file est une structure de données abstraite fondée sur le principe « premier arrivé, premier sorti ».
On munit la structure de données File des fonctions primitives suivantes.

Structure de données : File
Utilise : Booleen, Element
Opérations

- creer_file : ∅ →File

  creer_file() renvoie une file vide.

- est_vide : File→Booleen

  est_vide(f) renvoie True si la file f est vide et False sinon

- enfiler : File, Element→ Rien

  enfiler(f, e) ajoute l'élément e en queue de la file f

- defiler : File→Element

  defiler(f) renvoie l'élément qui est en tête de la file f et le retire de celle-ci.

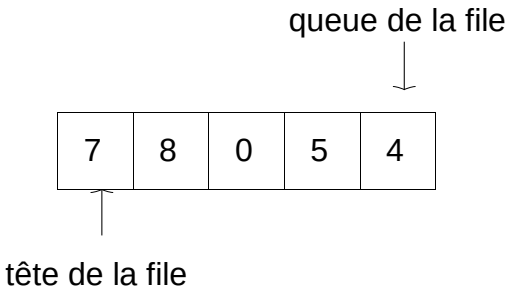


Figure 1 : une file

Partie A :

1. On suppose dans cette question que la file f est dans l'état de la *figure 1*.
   Quel est l'état de la file f après l'exécution des instructions suivantes?

```
1  for k in range(3) :
2      x = defiler(f)
3      enfiler(f, x)
```

2. On suppose dans cette question que la file f est dans l'état de la *figure 1*.

```
k = 0
g = creer_file()
while not est_vide(f) :
  x = defiler(f)
  k = k + 1
  enfiler(g, x)
while not est_vide(g) :
  x = defiler(g)
  enfiler(f, x)
```

Après l'exécution des instructions précédentes :

(a) Quel est l'état de la file f ?

(b) Quelle est la valeur de k ? Que représente la variable k ?

Partie B :

Une imprimante a une file d'attente d'impression dans laquelle sont placés les fichiers à imprimer. On peut modéliser cette file d'attente par une file dans laquelle les éléments sont des numéros permettant d'identifier les fichiers à imprimer. Ainsi tous les éléments dans la file sont différents deux à deux.

3. Dans la gestion de l'imprimante, on a la possibilité de supprimer un fichier de la file d'attente d'impression.

Voici le code incomplet de la fonction *supprimer* qui a pour paramètres une file f et un élément e présent une seule fois dans f et qui retire l'élément de la file. La fonction ne renvoie rien.

```
def supprimer(f, e) :
    g = creer_file()
    while not est_vide(f) :
        x = defiler(f)
#ligne à compléter
#ligne à compléter
    while not est_vide(g) :
#ligne à compléter
        enfiler(f, x)
```

Recopier et compléter le code de la fonction.

4. Dans la gestion de l'imprimante, une option permet d'imprimer un fichier en priorité. Ainsi le fichier est placé en tête de la file d'attente d'impression.

En modifiant la fonction *supprimer*, écrire une fonction *placer_en_priorite* qui a pour paramètres une file f et un élément e et qui déplace cet élément en tête de file. On suppose que l'élément e est déjà présent dans la file.

EXERCICE 2 (4 points)

Cet exercice porte sur les arbres binaires de recherche.

Les questions sont indépendantes.

Dans cet exercice, on s'intéresse aux arbres binaires de recherche dont les clés des nœuds sont des nombres entiers tous différents deux à deux.

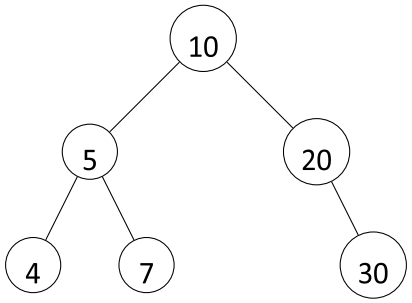


Figure 1 : un arbre binaire de recherche

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.

1. On rappelle que lors d'un parcours *en profondeur* d'un arbre binaire, on définit trois parcours des nœuds de l'arbre :
   - le parcours préfixe;
   - le parcours infixe;
   - le parcours suffixe (ou postfixe).

   On parcourt l'arbre de la *figure 1 en profondeur*.

   (a) Quel est le parcours suffixe de cet arbre?

   (b) Quel parcours permet d'avoir les clés des nœuds de l'arbre dans l'ordre croissant, c'est à-dire 4, 5, 7, 10, 20, 30 ?

2. On rappelle qu'un arbre ne comportant qu'un seul nœud a une hauteur égale à 1.

   (a) Recopier l'arbre binaire de recherche de la *figure 1* après l'insertion de la clé 28.

   (b) Quelle est la hauteur de l'arbre obtenu après cette insertion?

   Dans la suite, 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.
```

3. On définit une fonction *mystere* qui a pour paramètre un ABR a.

```
1  def mystere(a) :
2      assert not est_vide(a), 'L'arbre est vide'
3      if est_vide(sous_arbre_droit(a)) :
4        return racine(a)
5      return mystere(sous_arbre_droit(a))
```

(a) Quelle valeur est renvoyée lors de l'appel de la fonction *mystere* avec l'arbre de la *figure 1* comme argument ?

(b) Que représente cette valeur ?

4. Recopier et compléter la fonction *rechercher* qui a pour paramètres un ABR a et une clé e, et qui renvoie True si la clé e est dans l'arbre et False sinon.

```
1  def rechercher(a, e) :
2      if est_vide(a) :
3          return # A compléter
4      elif # A compléter :
5          return True
6      elif e < racine(a) :
7          return # A compléter
8      else :
9        return # A compléter
```

EXERCICE 3 (4 points)

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

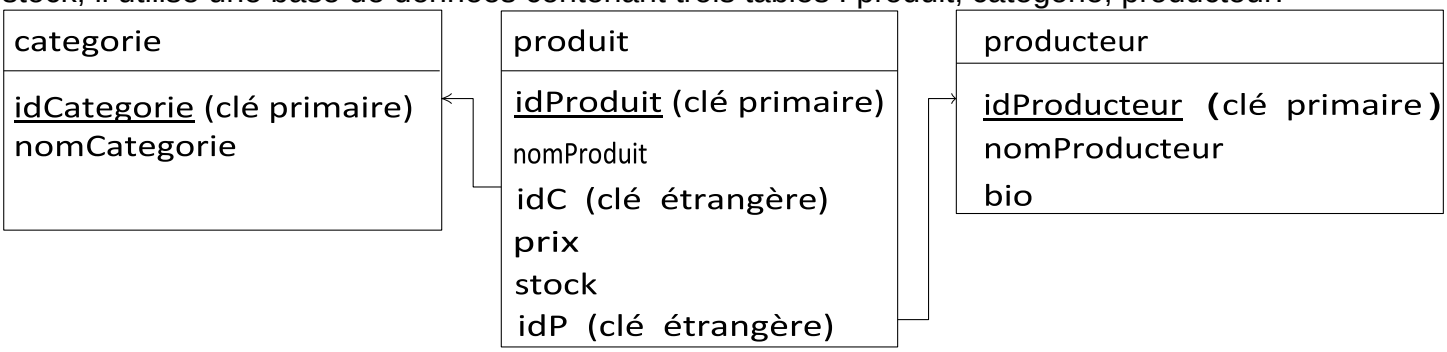
Dans cet exercice, on pourra utiliser les mots du langage SQL suivant :

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

La commande SUM permet de calculer la somme des valeurs d'une colonne en utilisant la syntaxe ci-dessous :

```
SELECT SUM (attribut) FROM table
```

Un primeur achète des fruits, des légumes et des fromages à des producteurs. Pour gérer son stock, il utilise une base de données contenant trois tables : produit, categorie, producteur.



Dans la table produit :

- L'attribut idC est une clé étrangère qui fait référence à la clé primaire **idCategorie** de la table categorie;
- L'attribut idP est une clé étrangère qui fait référence à la clé primaire **idProducteur** de la table producteur.

Vous trouverez en Annexe le contenu des trois tables.

Les questions sont indépendantes.

1. C'est la pleine saison des carottes. Le primeur décide de modifier le prix de celles-ci. Elles passent donc de 1,95 € à 1,80 €.
   Ecrire la requête en langage SQL permettant de modifier le prix de ce produit dont l'attribut idProduit vaut 1.

2. Le primeur veut rajouter un nouveau produit, le citron caviar, à la base de données. Il exécute la requête suivante :

```
INSERT INTO produit
 VALUES (112, "citron caviar", "Z", 155, 3, 6);
```

Que se passe-t-il? Justifier la réponse.

3. Le primeur souhaite regarder les produits dont le stock est peu important pour relancer les producteurs.
   (a) Ecrire une requête qui affiche tous les produits dont le stock est inférieur à 5 kg.
   (b) Que doit-on ajouter à la requête pour afficher les résultats précédents dont le prix est supérieur à 2€ ?

4. Le primeur souhaite connaître le poids de tout son stock. Quelle requête doit-il écrire ?

5. Lors d'un contrôle, le primeur doit fournir la liste de tous ses produits bio avec le nom du produit, l'identifiant du producteur, le nom du producteur. Quelle requête permet d'établir cette liste ?

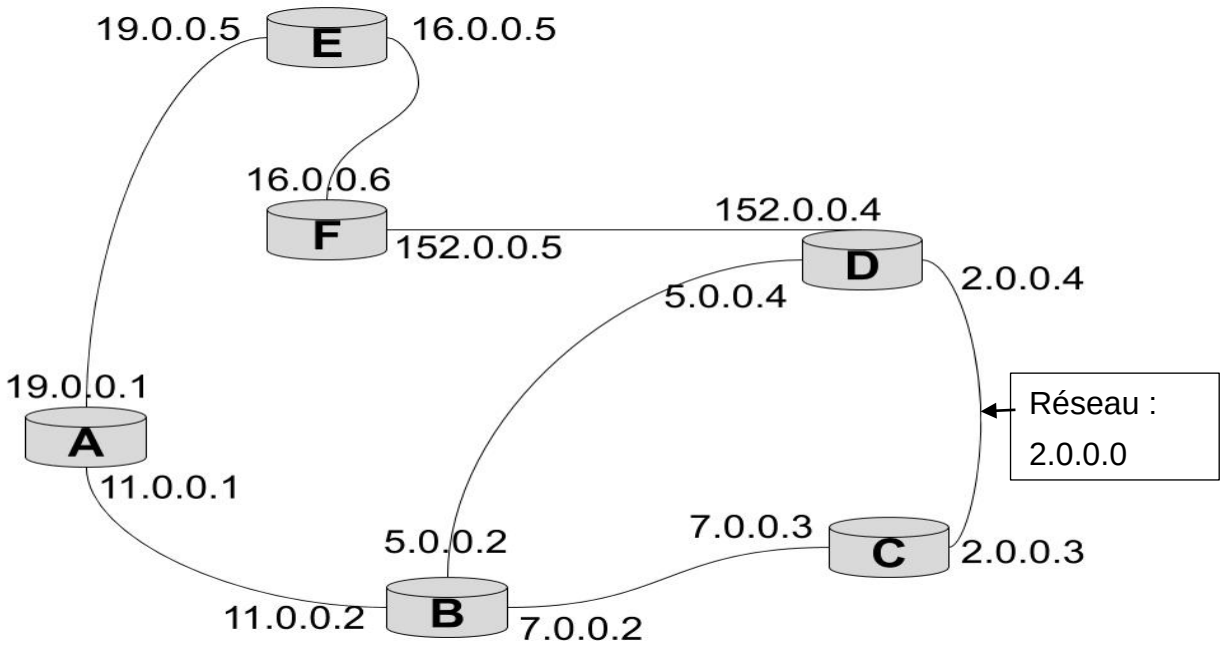
EXERCICE 4 (4 points)

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

Le schéma suivant explicite les connexions entre les routeurs A, B, C, D, E et F.

Pour chaque routeur, l’adresse IPv4 (version 4) de chacune de ses interfaces est précisée.

Ces adresses sont définies sur 4 octets, pour identifier chaque machine connectée sur le réseau informatique. On considère que le masque de sous-réseau est, dans tous les cas, 255.0.0.0.



PARTIE A : Architecture

On désire créer un réseau informatique équipé de 3 machines et connecté en filaire sur le routeur B. L’adresse IPv4 du nouveau réseau est 192.168.10.0, le masque de sous réseau est 255.255.255.0

1. Toutes les machines devront, pour se connecter à ce réseau, disposer d’une carte électronique dédiée. Donner son nom.
2. Quel appareil assure la connexion des machines sur ce réseau ?
3. Combien de machines pourront avoir une adresse IPv4 sur ce réseau ?
4. Donner l’adresse IPv4 pour le routeur B sur le nouveau réseau.
5. Donner une adresse IPv4 pour la machine 1.
6. Dessiner sur votre copie une représentation de ce réseau.

PARTIE B : Table de routage

7. Dans cette question, on utilise le protocole de routage **RIP** qui minimise le nombre de routeurs traversés.
Recopier et compléter les lignes incomplètes de la table de routage du routeur B ci-dessous :

Réseau de destination	Passerelle	Métrique
2.0.0.0	7.0.0.2	1
5.0.0.0	5.0.0.2	0
7.0.0.0	7.0.0.2	0
11.0.0.0	11.0.0.2	0
16.0.0.0		
19.0.0.0		
152.0.0.0		

8. Dans la question suivante, on utilise le protocole de routage **OSPF** qui minimise le coût des liaisons. On considère :

- que le coût d'un contact établi entre une machine et l'interface d'un routeur d'un même réseau est zéro ;
- que le coût de la traversée d'un réseau est donné par le tableau suivant :

Réseau	19.0.0.0	11.0.0.0	16.0.0.0	5.0.0.0	152.0.0.0	7.0.0.0	2.0.0.0
Coût	10	10	1	1	1	10	1

Déterminer la route suivie si les informations partent du routeur B et se dirigent vers le routeur E. Justifier votre réponse après avoir calculé le coût total de chaque parcours possible.

9. Si la liaison du réseau 5.0.0.0 est coupée, quel est le nouveau chemin proposé par la table de routage OSPF pour le même parcours des informations ?

EXERCICE 5 (4 points)

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

Le Scrabble est un jeu de société qui se joue à plusieurs joueurs. Chaque joueur pioche des jetons sur lesquels sont inscrites une lettre majuscule et sa valeur.

Le joueur propose ensuite un mot qu'il dispose sur un plateau de jeu en plaçant un jeton par case.

Certaines cases permettent une bonification.

1. Lorsqu'on pose un mot sur des cases ne présentant aucune bonification, on calcule le score en additionnant la valeur des lettres qui composent le mot.

J 8	E 1	U 1

vaut 10 points.

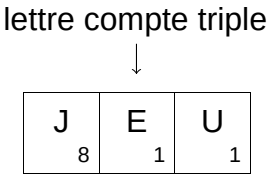
Voici le code d'une fonction *calculer_score_sans_bonif* qui a pour paramètre une chaîne de caractères mot et qui renvoie le score du mot lorsqu'il n'y a pas de bonification.

```
def calculer_score_sans_bonif(mot) :
1     scrabble = {"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
2             "H":4, "I":1, "J":8, "K":10,"L":1,"M":2,"N":1,"O":1,
3             "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
4             "X":10,"Y":10, "Z":10}
5     score = 0
7        for k in range(len(mot)) :
8    score = scrabble[mot[k]]
9    return score
```

Une erreur s'est glissée dans le corps de la fonction entre les lignes 5 et 9 incluses.
Indiquer la ligne à modifier et corriger l'erreur.

2. Sur certaines cases est indiquée la bonification "lettre compte double" ou "lettre compte triple".

Exemple :



Le score obtenu est de 12.

La fonction *calculer_score_avec_bonif* a pour paramètres :

- une chaîne de caractères mot
- une liste bonif de même longueur que mot précisant pour chaque lettre la bonification correspondante.

Cette fonction renvoie le score du mot en tenant compte des bonifications.

Dans l'exemple précédent on appelle *calculer_score_avec_bonif* ("JEU", [1, 3, 1]) et on obtient 12.

```
1 def calculer_score_avec_bonif(mot, bonif) :
2     scrabble={"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
3    "H":4, "I":1, "J":8, "K":10, "L":1,"M":2,"N":1,"O":1,
4     "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
5     "X":10,"Y":10, "Z":10}
6     score = 0
7     for k in range(len(mot)) :
8          score = # A compléter
9     return score
```

Recopier et compléter la ligne incomplète.

3. Lucien joue une partie avec ses grands-parents. Avec le tirage B A O L E L N, il a composé une liste de mots : BAL, NOBLE, ANE, BALLE, LOBE. Il souhaite mettre la lettre B sur une case "lettre compte triple", mais pour cela le B doit être la 3e lettre du mot qu'il doit poser.

(a) Ecrire une fonction *mot_lettre_position* qui a pour paramètres :

- une liste de mots liste
- un caractère c
- une position i (la position 2 correspond à la 3e lettre)

et qui renvoie la liste de tous les mots qui sont dans liste et qui ont le caractère c en position i.

(b) Quel appel Lucien doit-il effectuer pour trouver les solutions à son problème?

4. Pour passer le temps entre deux tours, Lucien décide de chercher les anagrammes d'un mot. L'anagramme d'un mot est un nouveau mot formé en changeant de place les lettres du mot initial.

Ainsi CARTE est une anagramme du mot ACTER.

On se propose d'écrire une fonction récursive *anagramme* qui a pour paramètres deux chaînes de caractères mot1 et mot2 et qui renvoie True si mot2 est une anagramme de mot1 et False sinon.

Pour cela, on dispose d'une fonction *enlever* :

```
enlever (mot , c)  renvoie mot privé de la première
occurence du caractère c
```

Ainsi l'instruction `enlever('scrabble','b')` renvoie 'scrable'.

```
1   def anagramme(mot1, mot2):
2     if len(mot1) != len(mot2):
3         return # A compléter
4     elif len(mot1) == 0:
5         return # A compléter
6     elif not mot1[0] in mot2 :
7         return # A compléter
8     else :
9         # Partie récursive à compléter
```

Recopier et compléter la fonction récursive anagramme.

Annexe de l'exercice 3

categorie	
idCategorie	nomCategorie
A	Fruit
B	Légume
C	Fromage

producteur		
idProducteur	nomProducteur	bio
1	Laiterie du Haut Cantal	1
2	La ferme de Pierre	1
3	La ferme du haut plateau	0
4	Le verger de Julie	1
5	Patrick Berger	0
6	La ferme d'Emile	0

produit					
idProduit	nomProduit	idC	prix	stock	idP
1	carotte	B	1,95	10	3
2	poireau	B	5,05	8	4
3	pomme	A	2,60	12	2
4	pomme	A	1,55	4	3
5	tomme de vache	C	23,90	6	5
6	tomme de brebis	C	31,50	3	1
		...			
		...			
		...			
		...			
		...			
111	pomme de terre	B	1,25	15	4

dernier enregistrement →

- Les prix sont en €.
- Les stocks sont en kg.
- bio vaut 1 lorsque le producteur est en agriculture biologique et 0 sinon.