MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

EBE NSI 2

SESSION 2022

CONCOURS EXTERNE
CAPES ET CAFEP/CAPES CORRESPONDANTS

Section

NUMERIQUE ET SCIENCES INFORMATIQUES

ÉPREUVE DISCIPLINAIRE APPLIQUEE

Durée : 5 heures

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout matériel électronique est rigoureusement interdit.

- *Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence.*
- *De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.*

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.

Tournez la page S.V.P.

Épreuve disciplinaire appliquée

Préambule : cette épreuve est constituée de deux problèmes indépendants. Les réponses aux questions devront être précises et rédigées avec soin.

Problème 1 - Gestion des données

1 Introduction à la problématique des données

Vous souhaitez aborder la notion de données et l'importance que les données ont pris dans notre société avec vos élèves de SNT en seconde.

1. Citer un exemple que vous pourriez utiliser devant les élèves pour illustrer l'importance de l'exploitation des données dans leur vie quotidienne. Vous justifierez brièvement votre choix.
2. Expliquer, en quelques lignes et comme vous l'expliqueriez à vos élèves, pourquoi le stockage de données sur le « cloud » (donc dans les datacenters) a bien un impact négatif sur l'environnement. Vous donnerez deux impacts différents sur l'environnement.
3. Donner deux conseils concrets que vous pourriez donner à vos élèves pour limiter facilement l'impact de leurs données sur l'environnement.
4. Expliquer en quoi disposer de données massives (appelées aussi Big Data) peut être un avantage dans certains enjeux de société. Illustrer votre réponse avec un exemple que vous pourriez proposer à vos élèves.
5. Citer un danger (autre que l'impact sur l'environnement) qui est apparu ou peut apparaître avec les données massives. Vous expliquerez, comme vous l'expliqueriez à vos élèves, ce danger en quelques lignes.

2 Utilisation des fichiers csv

Vous voulez maintenant traiter la notion de données structurées toujours avec vos élèves de seconde.

6. Définir la notion de descripteur d'objet.
7. Proposer une activité débranchée que vous pourriez réaliser en classe pour leur faire comprendre la notion de descripteur d'objet. Cette activité devra durer entre 30 min et 1h. On attend une description détaillée du déroulement de l'activité et de ses attendus.

Vous abordez ensuite le traitement de ces données structurées. Voici l'extrait du programme de seconde SNT sur ce point :

Contenus	**Capacités attendues**
Traitement de données structurées	Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables

Pour cela, vous décidez de les faire travailler sur le fichier csv suivant : « Donnees_capteurs_22_09_21.csv » (cf Annexe A). Ce fichier contient toutes les mesures réalisées par les capteurs d'une maison domotisée le 22/09/2021. Chaque capteur envoie ses mesures toutes les 5 minutes. Ce sont des capteurs polyvalents placés sur les ouvertures d'une maison (fenêtres/portes) qui permettent de détecter l'ouverture ou non mais aussi le mouvement et la température. Les différentes colonnes disponibles dans ce fichier sont :

- Identifiant_capteur (entier) : identifiant unique du capteur.
- Piece (texte) : pièce dans laquelle se trouve le capteur.
- Num_mesure (entier) : numéro de la mesure de la journée pour le capteur (1 à minuit et s'incrémente à chaque mesure jusqu'à la fin de la journée).

- Ouverture (booléen) : vaut True si la porte/fenêtre est ouverte et False sinon.
- Presence (booléen) : vaut True si le capteur détecte un mouvement et False sinon.
- Temperature (flottant) : indique la température mesurée à 0.1°C près.

Le fichier csv est pour l'instant ouvert par les élèves dans un tableur comme Calc/Excel.

8. Donner 3 questions que vous pourriez poser aux élèves pour les faire travailler, sur ordinateur, sur le traitement de données structurées, le but étant ici de leur faire découvrir les traitements possibles. Vous expliquerez à chaque fois brièvement l'intérêt de la question pour l'acquisition de compétence des élèves. Chaque question devra faire travailler les élèves sur une opération différente.

Vous souhaitez maintenant les faire travailler sur le traitement du fichier à l'aide de Python en utilisant la bibliothèque pandas. Vous trouverez en annexe une description des fonctions utiles de pandas (cf Annexe B).

Pour une exploitation du fichier « Donnees_capteurs_22_09_21.csv » avec la bibliothèque pandas, on s'est assuré que les nombres réels sont désormais écrits avec des points et non des virgules.

Le but de votre exercice est que les élèves puissent répondre de manière automatisée grâce à Python aux questions suivantes (pour le jour considéré) :

- Donner la température minimale de la cuisine quand le capteur 3 indique la présence de quelqu'un.
- Afficher toutes les températures de la maison triées de manière croissante.

9. Écrire, tel qu'un élève pourrait le faire, le programme permettant de répondre à ces questions à l'aide de la bibliothèque pandas.
10. Décrire la structure d'une activité de 1h30 sur machine permettant à vos élèves, qui n'ont jamais utilisé pandas, d'être capables de répondre à ces questions. On ne demande pas ici de rédiger complètement le document qui serait fourni aux élèves mais d'indiquer clairement l'organisation de l'activité en précisant notamment les questions posées aux élèves.

3 Traitement de données sous forme de tables

Vous travaillez sur le traitement de données en tables avec vos élèves de première. Pour travailler sur les données en tables, vous avez choisi de travailler avec des tableaux de p-uplets. Avec les élèves, les p-uplets seront codés, en Python, à l'aide de dictionnaires. Dans cet objectif, vous préparez un cours sur les dictionnaires.

11. Présenter, tel que vous le feriez avec vos élèves, la différence entre le type dictionnaire et le type liste.
12. Expliquer brièvement comment les dictionnaires sont implémentés en Python.
13. Citer un exemple d'utilisation de dictionnaire pour illustrer votre cours sur les dictionnaires. Vous justifierez votre choix en expliquant notamment pourquoi l'exemple est pertinent pour ce cours de première. Qu'est-ce qui pourrait être un mauvais exemple d'un point de vue pédagogique et pourquoi ?

Vous décidez de travailler avec vos élèves à partir de deux fichiers csv présentés en annexe pour démarrer le traitement des données en tables (annexe C). Pour que les élèves comprennent bien les mécanismes en jeu, vous n'utilisez pas ici la bibliothèque Python pandas.
Vous vous êtes inspirés de la situation sanitaire actuelle pour créer deux fichiers (qui ne sont bien-sûr que des fichiers exemples) : l'un correspondant à la liste des personnes testées positives à la Covid19 et l'autre à la liste des personnes ayant mangé au restaurant. Ces fichiers sont supposés être stockés par l'assurance maladie, le second servant à contacter les personnes contacts à risques.

Le premier fichier nommé « positifs.csv » (cf Annexe C.1) contient les informations suivantes :

- le numéro de sécurité social du contaminé (descripteur : num_SS ; type : entier),
- son âge (descripteur : age ; type : entier),
- le jour du test positif (descripteur : jour_test ; type : entier),
- le mois du test positif (descripteur : mois_test ; type : entier),
- son sexe (descripteur : sexe ; type : caractère),
- s'il a été vacciné ou non (descripteur : vaccine ; type : booléen).

Le deuxième fichier nommé « restaurants.csv » (cf Annexe C.2) contient les informations suivantes :

- l'identifiant de l'enregistrement (descripteur : id_enregistrement ; type : entier),
- l'identifiant du restaurant (descripteur : id_restaurant ; type : entier),
- le jour du repas (descripteur : jour_repas ; type : entier),
- le mois du repas (descripteur : mois_repas ; type : entier),

Tournez la page S.V.P.

- la tranche horaire du repas (descripteur : tranche_horaire ; type : entier),
- le numéro de la table (descripteur : num_table ; type : entier),
- le numéro de sécurité social du client (descripteur : num_SS ; type : entier).

Pour cet exercice, on ne considèrera que 2 tranches horaires (la 1 de 12h à 14h et la 2 de 19h à 21h) et on considérera qu'une même personne ne peut pas être testée deux fois positive à la Covid19.

14. Écrire le programme Python commenté qui permettrait à vos élèves de récupérer sous forme de liste de dictionnaires, nommée **positifs**, les informations contenues dans le fichier « positifs.csv ». Chaque élément de la liste est un dictionnaire comprenant les valeurs d'une ligne de la table et dont les clés sont les descripteurs de la table. On utilisera pour cela notamment la fonction readlines.
15. Quels sont pour vous les points importants à aborder avec vos élèves et/ou ceux qui risquent de leur poser problème lorsque vous abordez pour la première fois le programme de la question 14 avec vos élèves ?

Vous souhaitez faire travailler vos élèves sur la recherche, dans une table, des lignes vérifiant des critères exprimés en logique propositionnelle en utilisant les fichiers décrits ci-dessus. On supposera, pour la suite de l'exercice, que les élèves ont implémenté le programme Python retournant une liste de dictionnaires **restaurants** contenant les informations contenues dans le fichier « restaurants.csv ».

16. Vous rédigez maintenant une partie de cet exercice.
    (a) Proposer 3 questions de difficulté croissante (dont au moins une nécessitant la fusion des tables) où les élèves doivent rédiger des programmes en Python. On suppose ici que les listes de dictionnaires **positifs** et **restaurants** sont déjà construites.
    (b) Proposer pour chaque question une correction. Décrire brièvement les points sur lesquels vous serez vigilant lors de la correction de ces questions.

4 Utilisation des bases de données

Dans cette partie, vous travaillez sur les bases de données avec vos élèves de terminale.

17. Expliquer tel que vous le feriez avec vos élèves l'intérêt des bases de données par rapport à ce que les élèves ont fait en première (avec les tables sous forme de fichiers csv).

Voici un extrait du programme de NSI de terminale.

Contenus	Capacités attendues	Commentaires
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

Vous avez choisi, pour travailler sur le langage SQL, un exercice avec une base de données simplifiée qui stocke les données d'un service proposant des films en streaming. La base de données contient les tables suivantes :

Films ={Id, Titre, Annee, Pays}
Clients= {Id, Nom, Prenom, AdresseMail, TypeAbonnement}
Visionnage={Id, IdClient, IdFilm, Date, Heure, TempsVisionnage}
Facturation={Id, IdClient, Montant, Date, Statut}

Un extrait de cette base de données est fournie dans l'annexe D. Le temps de visionnage est stocké en minutes. Il existe deux types d'abonnement : le "basic" et le "premium". Les factures peuvent être "Payée" ou "En attente".

18. Voici l'exercice que vous avez donné aux élèves pour travailler sur les requêtes en SQL :
    (a) Écrire la requête SQL permettant d'ajouter le film suivant à la base de données : « Wicked » un film réalisé aux USA et sorti en 2021. Son identifiant est 20356.
    (b) Écrire la requête SQL permettant d'indiquer que la facture d'identifiant 2564 a été payée.
    (c) Écrire la requête SQL permettant d'obtenir la liste des identifiants de facturation qui sont en attente et qui ont un montant strictement supérieur à 5 euros.

(d) Écrire la requête SQL permettant d'obtenir la liste, par ordre alphabétique, des pays d'origine des films de la plateforme de l'année 2015. Chaque pays ne sera bien sûr indiqué qu'une seule fois.

(e) Écrire la requête SQL permettant d'obtenir la liste des noms, prénoms et adresses mails des clients ayant une facturation en attente.

(f) Écrire la requête permettant de déterminer le temps moyen de visionnage du film Titanic.

(g) Écrire la requête permettant de donner la liste des identifiants des clients qui ont regardé Titanic plus longtemps que la moyenne des utilisateurs.

Donner les réponses que vous pourriez attendre pour chacune de ces questions.

19. Des élèves peinent à rédiger la requête d'interrogation de données de la question 18(e). Quelle aide leur proposez-vous ?
20. Donner un barème pour la correction de la question 18(g) Vous détaillerez les différentes compétences qu'il vous semble important d'évaluer.
21. Proposer, à l'aide de cette base de données, 3 questions qui vous permettraient de valider les compétences de vos élèves sur les requêtes SQL lors d'une interrogation. Vous justifierez, pour chaque question, en quoi elle est pédagogiquement intéressante en indiquant notamment la (ou les) compétence(s) évaluée(s).

Problème 2 - Algorithmes de tri

Trier une collection de données avant de résoudre un problème algorithmique permet souvent d'élaborer des solutions beaucoup plus efficaces. Par exemple, on peut effectuer une recherche dichotomique au lieu d'effectuer un parcours linéaire dans une collection triée ou le tri d'un nuage de points permet de calculer efficacement l'enveloppe convexe de ce nuage de points (parcours de Graham).

De nombreux algorithmes de tri procèdent par comparaisons successives d'éléments entre eux. Pour évaluer l'efficacité de ces algorithmes et les comparer, on évalue souvent le nombre de comparaisons effectuées lors de leur exécution : on parle alors de **complexité en nombre de comparaisons**. Si l'on considère d'autres opérations élémentaires telles que les affectations, on dit que l'on évalue la **complexité temporelle**. Dans ce sujet, en réponse aux questions de complexité, on attend des ordres de grandeur en notation Landau (par exemple $O(n \log n)$ ou $O(n^2)$ si n est le nombre d'éléments à trier). Certains algorithmes de tri peuvent avoir des ordres de grandeur différents pour la complexité en nombre de comparaisons et la complexité temporelle. Nous garderons cela à l'esprit lors de l'analyse des algorithmes de tri de ce sujet. Enfin, on rappelle qu'un algorithme de tri par comparaisons correct nécessite au moins $n \log n$ comparaisons dans le pire cas. Si nécessaire, vous pourrez vous référer à cette borne considérée comme admise par la suite.

La dernière section de cette partie concerne des algorithmes de tri qui n'utilisent pas de comparaison, nous évaluerons alors seulement la complexité temporelle.

5 Tris naïfs

Dans cette section, deux algorithmes de tri par comparaisons du programme de première NSI sont considérés : le tri par insertion et le tri par sélection. Ces tris sont dits naïfs car leur complexité n'est pas optimale.

5.1 Tri par insertion

22. Quatre élèves de première NSI vous ont rendu leur implémentation du tri par insertion. Pour chaque production, rédiger une appréciation concise, donnant les erreurs éventuelles et les améliorations possibles afin que les élèves puissent se corriger par elles-mêmes.

Tournez la page S.V.P.

```
def tri_eleve1(t) :
    n = len(t)
    for i in range(n) :
        for k in range(i,1,-1) :
            if t[k]<t[k-1] :
                t[k]=t[k-1]
                t[k-1]=t[k]
```

```
def insere_eleve2(t,k):
    if t[k]<t[k-1] :
        t[k],t[k-1]=t[k-1],t[k]
        insere_eleve2(t,k-1)

def tri_eleve2(t,i=0) :
    insere_eleve2(t,i)
    tri_eleve2(t,i+1)
```

```
def tri_eleve3(t):
    n = len(t)
    def insere_eleve3(i) :
        k = i
        while t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
            k = k-1
    for i in range(n) :
        t = insere_eleve3(i)
```

```
def tri_eleve4(t) :
    n = len(t)
    i = 0
    while i < n :
        k = i
        while k > 0 or t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
        i = i + 1
```

23. Pour que chacune puisse apprendre des erreurs des autres, rédiger une synthèse succincte à destination des élèves de première NSI dont vous venez de corriger les productions. L'idée est de s'appuyer sur les erreurs observées pour faire des rappels généraux sur les points clés à vérifier dans un programme afin qu'il fonctionne correctement et efficacement.
24. Une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour améliorer la complexité de l'insertion des éléments. Que lui répondez-vous ?

5.2 Tri par sélection

25. On rappelle succinctement le fonctionnement du tri par sélection au travers du pseudo-code suivant :

```
def tri_sel(t) :
    pour k de 0 à n-1
        sélectionner l'indice i0 du minimum des éléments entre les indices de k à n-1
        échanger t[k] et t[i0]
```

Écrire l'énoncé d'un devoir maison permettant de faire étudier cet algorithme à vos élèves de première NSI. Cet énoncé devra les guider pour :

- découvrir cet algorithme,
- l'implémenter,
- prouver sa correction,
- estimer sa complexité en nombre de comparaisons.

Attention, le contenu de cet énoncé devra se suffire à lui-même, et ne dépendre d'aucune ressource externe ou interventions spécifiques de votre part.

26. À la fin de la séance, une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour la sélection des éléments. Que lui répondez-vous ?

6 Tris par fusion

Cette section s'articule autour du tri partition-fusion du programme de terminale NSI et d'une version simplifiée du tri *Timsort* qui agit également par fusions successives.

6.1 Tri partition-fusion

Lors d'un devoir sur table en classe de terminale NSI, vous avez demandé à vos élèves d'implémenter le tri partition-fusion sans plus de précision. Trois productions d'élèves sont reproduites ci-dessous.

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])
```

```
def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res
```

```
def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]
```

```
def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
    aux(0,len(t))
```

27. Élaborer un barème détaillé pour cette question.

28. Corriger ces trois productions, c'est-à-dire donner la note détaillée selon le barème et l'appréciation associée pour chacune des productions d'élèves. **Les corrections seront données directement sur les productions d'élèves fournies dans le document réponse 1 (Annexe E). Ce document réponse est à rendre avec votre copie.**

Tournez la page S.V.P.

6.2 Tri *Timsort* simplifié

On présente ci-dessous une implémentation d'un tri *Timsort* simplifié.

```
def f1(l,a):
    i = a
    while i<len(l)-1 and l[i]<=l[i+1] :
        i += 1
    return i+1

def f2(l):
    i = 0
    res = [0]
    while i<len(l) :
        i = f1(l,i)
        res.append(i)
    return res

def f3(t, debut, milieu, fin) :
    i = debut
    j = milieu
    lt = []
    while i < milieu and j < fin :
        if t[i] < t[j] :
            lt.append(t[i])
            i += 1
        else :
            lt.append(t[j])
            j += 1
    for k in range (i, milieu) : lt.append(t[k])
    for k in range (debut, j) : t[k] = lt[k-debut]

def f4(t,rc):
    nrc = []
    i = 0
    n = len(rc)
    while i < n :
        nrc.append(rc[i])
        if i+2 < n :
            f3 (t, rc[i], rc[i+1], rc[i+2])
            i += 2
        else : i += 1
    return nrc

def f5(t):
    rc = f2(t)
    while len(rc) > 2 :
        rc = f4 (t, rc)
```

29. (a) Proposer des noms explicites pour ces fonctions.
    (b) Présenter le fonctionnement général de ces fonctions tel que vous le feriez à vos élèves.
    (c) Illustrer le principe de ce tri sur un ou plusieurs exemples pour que les élèves puissent l'implémenter par elles-mêmes.
30. Quelle est la complexité, dans le pire cas, en nombre de comparaisons de cet algorithme de tri ?

7 Tris sans comparaison

Votre classe de terminale NSI a bien compris les différents algorithmes du tri du programme. Vous décidez donc d'aller plus loin en leur faisant découvrir d'autres algorithmes de tri qui fonctionnent sans comparaison sur des listes d'entiers positifs.

7.1 Tri par dénombrement

Au lieu de comparer les éléments entre eux, le tri par dénombrement compte le nombre d'occurrences de chaque élément. Une fois les nombres d'occurrences obtenus, la liste est reconstituée dans l'ordre croissant en y plaçant tous les éléments présents dans l'ordre croissant avec le bon nombre d'occurrences.

Par exemple, pour trier $[1, 7, 5, 3, 7, 7, 1, 3, 3, 7, 5]$, on compte qu'il y a :

— 2 occurrences de 1

— 3 occurrences de 3

— 2 occurrences de 5

— 4 occurrences de 7

On peut alors reconstituer la liste triée : $[1, 1, 3, 3, 3, 5, 5, 7, 7, 7, 7]$.

Vous avez posé la question suivante à vos élèves :

Écrire la fonction Python `tri_den` *qui prend en paramètre une liste* `t` *d'entiers positifs, ne renvoie aucune valeur mais y applique le tri par dénombrement. La fonction proposée aura une complexité temporelle en* $O(n + vmax)$ *où* n *est le nombre d'éléments dans* `t` *et vmax est la valeur maximale dans* `t`.

Une élève vous rend le travail suivant :

```
def compte_occu(t,elt):
    c = 0
    for x in t :
        if x == elt : c = c+1
    return c

def tri_den_el(t):
    occ = [0]*10000
    res = []
    for elt in range(10000):
        occ[elt] = compte_occu(t,elt)
        res = res + [elt]*occ[elt]
    return res
```

31. Lister les indications à lui donner pour qu'elle puisse répondre correctement à votre question.
32. Écrire, sans commenter, le programme que vous espérez qu'elle écrive.
33. La complexité temporelle de cet algorithme contredit-elle la borne de complexité admise en introduction de ce sujet ? Pourquoi ?
34. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n \log n)$? Justifier.

7.2 Tri par baquets

Dans cette partie, le tri par baquets est étudié, le principe est le suivant :

- on travaille avec des baquets numérotés de 0 à 9 (ces baquets sont implémentés par une liste de 10 listes) ;
- pour débuter, chaque élément de la liste initiale est placé dans le baquet numéroté par son chiffre des unités ;
- à chaque étape, les éléments sont parcourus baquet par baquet en partant de l'indice 0 du baquet 0 et rangés à nouveau dans les baquets en utilisant le chiffre suivant ;
- l'algorithme est terminé lorsque tous les éléments sont dans le baquet 0.

Par exemple : si

$$\texttt{t} = [8295, 7971, 8806, 1203, 2916, 6231, 1112, 6131, 538, 3832, 1813, 6863, 6200, 9449, 1905, 1749]$$

- au départ, les éléments sont rangés dans les baquets par chiffres des unités :

$$[[6200], [7971, 6231, 6131], [1112, 3832], [1203, 1813, 6863], [], [8295, 1905], [8806, 2916], [], [538], [9449, 1749]]$$

Tournez la page S.V.P.

- puis on les place par chiffre des dizaines en préservant l'ordre sur les unités dans chaque baquet grâce à l'ordre de parcours des éléments :

$$[[6200, 1203, 1905, 8806], [1112, 1813, 2916], [], [6231, 6131, 3832, 538], [9449, 1749], [], [6863], [7971], [], [8295]]$$

- puis par chiffre des centaines en respectant l'ordre précédent dans chaque baquet :

$$[[], [1112, 6131], [6200, 1203, 6231, 8295], [], [9449], [538], [], [1749], [8806, 1813, 3832, 6863], [1905, 2916, 7971]]$$

- puis par chiffre des milliers :

$$[[538], [1112, 1203, 1749, 1813, 1905], [2916], [3832], [], [], [6131, 6200, 6231, 6863], [7971], [8295, 8806], [9449]]$$

- à la dernière étape, tous les éléments se retrouvent dans le baquet 0 :

$$[[538, 1112, 1203, 1749, 1813, 1905, 2916, 3832, 6131, 6200, 6231, 6863, 7971, 8295, 8806, 9449], [], [], [], [], [], [], [], [], []]$$

35. Écrire une fonction Python `ch` qui prend en paramètre deux entiers et telle que l'appel `ch(k,i)` renvoie le `i`-ème chiffre de l'entier `k`. Par exemple, `ch(1345,2) = 4`, `ch(1345,4) = 1` et `ch(1345,5) = 0`. Cette fonction pourra être utilisée par vos élèves pour l'implémentation du tri par baquets.
36. Vous avez fourni une implémentation à trous de l'algorithme de tri par baquets à vos élèves. En particulier, la fonction Python `tri_baquets` prend en paramètre une liste `t`, applique le tri par baquets à `t` et renvoie une liste triée contenant les mêmes éléments que `t`.

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[............].append(............)

def vider(bacs):
    for i in range(10):
        bacs[i] = ............

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[............].append(............)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = ............
        vider(............)
        i += 1
    return ............
```

Compléter ce que vous attendez que les élèves écrivent dans les trous sans commenter. **Vous compléterez cet algorithme à trous fourni dans le document réponse 2 (Annexe F). Ce document réponse est à rendre avec votre copie.**

37. Quelle est la complexité temporelle de ce tri ?
38. Dans quels cas utiliser cet algorithme plutôt que le tri par dénombrement ?
39. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n \log n)$?

A Extrait du fichier Donnees_capteurs_22_09_21.csv (ouvert dans un tableur)

Le séparateur du fichier .csv est " ;".

	A	B	C	D	E	F
1	Identifiant_capteur	Piece	Num_mesure	Ouverture	Presence	Temperature
2	1	Salon	1	False	False	18,7
3	3	Cuisine	1	False	False	18,8
4	2	Chambre 1	1	True	False	17,5
5	5	Salle de bair	1	False	False	19,1
6	6	Chambre 2	1	False	True	16,6
7	4	Entree	1	False	False	17,4
8	1	Salon	2	False	False	18,7
9	3	Cuisine	2	False	False	18,8
10	2	Chambre 1	2	True	False	17,5

B Description de fonctions utiles de la bibliothèque pandas

- pandas.read_csv(nom,sep=...) prend en argument une chaîne contenant l'adresse du fichier csv à lire et un chaîne contenant le séparateur du fichier csv, et retourne une instance de type dataframe, initialisée avec l'ensemble des données présentes dans le fichier. Lors de la création du dataframe, un index est associé à chacune des lignes du fichier (en commençant à partir de 0).

Par la suite, pour illustrer l'utilisation de certaines fonctions, on suppose que la variable "df" pointe vers le dataframe obtenu à partir du fichier Donnees_capteurs_22_09_21.csv.

- df.loc(index_ligne,index_colonne) retourne, sous forme d'un objet de type dataframe, certaines lignes et colonnes du dataframe df. L'index des colonnes est le nom des colonnes dans le fichier csv, et l'index des lignes est l'index créé lors de l'initialisation du dataframe à la lecture du fichier.
  Par exemple, print(df.loc[0,'Temperature']) affichera la température de la 1ère ligne du fichier.
  Il est également possible d'effectuer des tests sur le contenu d'un dataframe.
  Par exemple, print(df['Piece']=='Salon') affichera, sous forme de booléens, pour chaque ligne du fichier, si la pièce mentionnée est un salon.
  Ces deux techniques peuvent être conjointement utilisées pour filtrer des lignes à partir de tests.
  Par exemple, print(df.loc[df['Piece']=='Salon', :]) affichera toutes les lignes qui concernent le salon.
  Il est possible de combiner plusieurs facteurs de filtrage en utilisant un "et"(&) ou un "ou"(|).
  Par exemple, print(df.loc[(df['Piece']=='Salon')|(df['Piece']=='Cuisine'),'Presence']) affichera la colonne présence pour les lignes correspondant au salon ou à la cuisine.

- Il est possible de retourner le maximum (méthode max), le minimum (méthode min) et la moyenne (méthode mean) d'une donnée d'une dataframe.
  Par exemple, print(df.loc[:,'Temperature'].mean()) affichera la température moyenne mesurée sur l'ensemble des lignes du fichier.

- Pour retourner le contenu trié d'un dataframe, on utilise la méthode dataframe.sort_values(column,ascending=...). column est une chaîne indiquant la colonne utilisée pour le tri. Si ascending vaut True (valeur par défaut), le tri est croissant, s'il vaut False, il est décroissant.
  Par exemple, df.sort_values(by=["Piece"], ascending=False) affiche les lignes du fichier triées par Piece dans l'ordre alphabétique inversé (en commençant par Z).

Tournez la page S.V.P.

C Extraits des fichiers pour le traitement des données sous forme de table

C.1 Extrait du fichier positifs.csv

```
num_SS;age;jour_test;mois_test;sexe;vaccine
1630580265006;58;2;7;H;True
2750846580156;46;2;7;F;False
1950412410023;26;3;7;H;False
2570235820132;64;3;7;F;False
```

C.2 Extrait du fichier restaurants.csv

```
id_enregistrement;id_restaurant;jour_repas;mois_repas;tranche_horaire;num_table;num_SS
1;324;30;6;2;5;1850659240032
2;324;30;6;2;5;2860372541115
3;796;1;7;1;8;1950412410023
4;796;1;7;1;8;1951012410056
5;796;1;7;1;8;1951232564031
```

D Extrait de la base de données

Films			
Id	***Titre***	***Annee***	***Pays***
12546	Entre les murs	2008	France
2506	Fight Club	1999	USA
67895	La cité de la peur	1994	France
54781	Yes Day	2021	USA

Clients				
Id	***Nom***	***Prenom***	***AdresseMail***	***TypeAbonnement***
1761	Dupont	Martin	dm@capes.fr	Basic
564	Tartanpion	Lucie	TLucie@mail.com	Premium
153	Durant	Hervé	RVDurant@capes.fr	Premium
789	Martin	Nina	Nina@pro.fr	Basic

Visionnage					
Id	***IdClient***	***IdFilm***	***Date***	***Heure***	***TempsVisionnage***
123456	1761	5468	11/08/2021	10 :45	35
93720	2456	14803	09/11/2020	21 :10	124
115986	1602	14972	27/03/2021	08 :30	2
78439	2395	6730	16/05/2020	22 :35	48

Facturation				
Id	***IdClient***	***Montant***	***Date***	***Statut***
2761	1254	7.90	12/07/2021	Payee
1065	1542	7.90	05/08/2020	Payee
1753	864	11.90	03/04/2021	En attente
1334	741	11.90	09/10/2020	Payee

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.
Ces codes doivent être reportés sur chacune des copies que vous remettrez.

► **Concours externe du CAPES de l'enseignement public** :

Concours	Section/option	Epreuve	Matière
EBE	62OOE	102	9312

► **Concours externe du CAFEP/CAPES de l'enseignement privé** :

Concours	Section/option	Epreuve	Matière
EBF	62OOE	102	9312

IMPRIMERIE NATIONALE – 22 0199 – D'après documents fournis

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

Né(e) le : / /

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR1

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

E Document Réponse 1 : Correction de trois productions d'élèves

Document réponse de la question 28 de la section 6.1 (Problème 2). **Ce document réponse doit être rendu avec votre copie.**

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])
```

```
def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res
```

```
def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]
```

```
def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
    aux(0,len(t))
```

Modèle CMEN-DOC v2 ©NEOPTEC

Nom de famille :
(Suivi, s'il y a lieu, du nom d'usage)

Prénom(s) :

Numéro Inscription :

(Le numéro est celui qui figure sur la convocation ou la feuille d'émargement)

Né(e) le : ☐☐/☐☐/☐☐☐☐

(Remplir cette partie à l'aide de la notice)

Concours / Examen : …….. **Section/Spécialité/Série :** ……

Epreuve : …….. **Matière :** …….. **Session :** ……..

CONSIGNES

- *Remplir soigneusement, sur CHAQUE feuille officielle, la zone d'identification en MAJUSCULES.*
- *Ne pas signer la composition et ne pas y apporter de signe distinctif pouvant indiquer sa provenance.*
- *Numéroter chaque PAGE (cadre en bas à droite de la page) et placer les feuilles dans le bon sens et dans l'ordre.*
- *Rédiger avec un stylo à encre foncée (bleue ou noire) et ne pas utiliser de stylo plume à encre claire.*
- *N'effectuer aucun collage ou découpage de sujets ou de feuille officielle. Ne joindre aucun brouillon.*

DR2

Tous les documents réponses sont à rendre, même non complétés.

NE RIEN ECRIRE DANS CE CADRE

F Document Réponse 2 : Algorithme de tri par baquets à trous à compléter

Document réponse de la question 36 de la section 7.2. **Ce document réponse doit être rendu avec votre copie.**

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[.....].append(.....)

def vider(bacs):
    for i in range(10):
        bacs[i] = .....

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[.....].append(.....)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = .....
        vider(.....)
        i += 1
    return .....
```