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

SUJET D'EXAMEN (SESSION OFFICIELLE)

Numérique et sciences informatiques - Session 2023

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 2023

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Jour 2

Durée de l'épreuve : **3 heures 30**

L'usage de la calculatrice n'est pas autorisé.

Le sujet comporte **8** pages numérotées de **1/8** à **8/8**.
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat traite les 3 exercices proposés

Le document réponse, page 8/8, est à rendre obligatoirement et dans son intégralité avec la copie.

Exercice 1 (4 points)

Dans cet exercice, on utilise le langage SQL (Structured Query Language).

Une association sportive « JUDOKA » demande à ses bénévoles d'assurer une permanence téléphonique du lundi au vendredi de 8h à 12h et de 14h à 17h.

Le président de cette association charge le secrétaire de créer une base de données qui stockera des informations concernant les bénévoles de cette association.

Le secrétaire utilise un système de gestion de base de données relationnelles (SGBDR) pour créer une base de données.

1) Le secrétaire utilise le code ci-dessous.

```
Benevole (idbenevole,nom,prenom,rue,code_postal,ville,telephone,#idmission)

Mission (idmission,designation)
```

Les clés primaires sont soulignées et les clés étrangères sont précédées du caractère #.

a. `idbenevole` est déclaré comme clé primaire de la table `Benevole`. **Préciser** son rôle.
b. `idmission` est déclaré comme clé étrangère de la table `Benevole`. **Préciser** son rôle.

2) Le secrétaire exécute la requête ci-dessous. **Expliquer** l'objectif de celle-ci.

```
INSERT INTO Benevole
     (nom, prenom, rue, code_postal, ville, telephone, idmission)
VALUES
        ('POHU', 'Dominique', '12 rue de la Frairie', '49000', 'Angers',
'+336123456', 1)
        ('MOISY', 'Camille', '33 rue Maurice Bot', '49400', 'Cholet',
'+337987654', 2) ;
```

3) Les bénévoles réalisent une vente de calendriers au profit de l'association.

Une table `Collecte` a été créée pour répertorier chaque année les ventes réalisées.

```
Collecte (idcollecte,#idbenevole,annee_ventes,montant)
```

- `idbenevole` fait référence à la table `Benevole` ;
- `annee_ventes` contient l'année lors de laquelle le bénévole apporte au trésorier sa collecte dont la valeur en euros est donnée par l'attribut montant. Il s'agit d'un nombre entier.

a. **Écrire** une requête SQL permettant d'afficher l'extrait de la table `Collecte` avec les ventes réalisées en 2020.
b. **Écrire** une requête SQL permettant d'afficher les noms et prénoms des bénévoles ayant apporté plus de 50 € de dons en 2020.
c. **Écrire** une requête SQL permettant de modifier une erreur de saisie sur le montant des ventes en 2020 par la personne dont l'id`benevole` est égal à 1. En effet, cette personne a collecté 300 € alors que le montant apparaissant dans la base est de 3 €.

4) Le secrétaire a créé la table suivante afin de suivre l'activité des bénévoles.

Presence (idpresence,jour,nb_heures)

a. Le secrétaire exécute la requête suivante pour insérer un premier enregistrement dans la table :

```
INSERT INTO Presence(idpresence, jour , nb_heures)
VALUES ('11-02-2020', 2) ;
```

Expliquer pourquoi cette requête échoue.

b. L'erreur de la question précédente a été corrigée. Le secrétaire exécute la requête suivante pour obtenir la liste des noms des bénévoles présents plus de 2 heures sur une journée :

```
SELECT nom FROM Presence WHERE nb_heures > 2 ;
```

Expliquer pourquoi cette requête échoue. **Proposer** une solution de correction.

Exercice 2 (4 points)

Tous les programmes de cet exercice seront écrits en langage Python.

1) Un mot est un palindrome lorsqu'il s'écrit de la même façon à l'endroit et à l'envers.
Exemples : radar, abcdcba, kayak, laval, non, ressasser, elle.
Soit la fonction ci-dessous appelée `palindrome`.

```
def palindrome(mot):
    n=len(mot)
    est_palindrome=True
    for k in range(n//2):
        if mot[k]!=mot[n-k-1]:
            est_palindrome=False
    return est_palindrome
```

a. **Écrire** les spécifications, sous la forme d'une chaîne de documentation (docstring), de cette fonction. Cette documentation précisera l'entrée et son type, la sortie et son type, le rôle de cette fonction.
b. **Indiquer** sans justification la complexité en temps de calcul de cet algorithme pour un mot de longueur n.
Remarque : on ne comptera que les comparaisons de caractères.
c. **Indiquer** ce qu'est une fonction récursive.

d. **Écrire et compléter**, sur la copie, les lignes 2, 4 et 5 de la fonction récursive appelée `palindrome_recursive`, qui prend en paramètre un mot et qui teste si ce mot est un palindrome.
e.

```
1. def palindrome_recursive(mot):
2.     if len(mot)<=……:
3.         return True
4.     if mot[0]!=mot[……]:
5.         return ……
6.     return palindrome_recursive(mot[1:len(mot)-1])
```

Remarque : l'expression `chaine[debut:fin]` donne la sous-chaîne comportant les caractères de l'indice `debut` inclus à l'indice `fin` exclu.

2) On s'intéresse à une version simplifiée du jeu télévisé Motus. Le jeu repose sur la recherche de mots d'un nombre fixé de lettres. Un candidat propose un mot et doit l'épeler. Le mot doit être correctement orthographié, sinon il est refusé.
On s'intéresse uniquement aux lettres bien placées. Un mot mystère est défini au préalable. Sa première lettre est donnée et ses lettres restantes signalées par des tirets. Le programme invite alors le candidat à saisir un mot.

Par exemple :
- Le mot mystère est MOMIE.
- Le programme va d'abord afficher « M---- ».
- Le candidat est ainsi invité à chercher un mot de cinq lettres commençant par M.
- Le candidat propose le mot MINCE.

Les lettres M et E sont bien placées dans le mot à deviner MOMIE, la console affiche M---E.

a. **Écrire** la fonction `test_mot` qui prend en paramètres le mot à deviner en majuscules et un mot proposé en majuscules par le joueur. Cette fonction renvoie le mot formé des lettres correctement placées et de tirets ailleurs comme dans l'exemple ci-dessus. On suppose que le mot proposé par le joueur a le bon nombre de lettres.

Par exemple :
```
>>>test_mot('MOMIE','MINCE')
'M---E'
```

b. On a rédigé un programme `jeu_motus` qui prend en paramètre le mot à deviner et qui :

- affiche la première lettre du mot à deviner suivie de tirets ;
- affiche la phrase « donner votre proposition : » qui invite le candidat à entrer une proposition ;
- affiche les lettres bien placées et des tirets ailleurs ;
- recommence jusqu'à ce que le candidat ait gagné ou qu'il ait atteint le nombre maximal d'essais autorisés, ce nombre maximal étant égal au nombre de lettres du mot ;
- affiche le message « Félicitations » si le candidat a gagné ;
- donne le mot mystère si le candidat a perdu.

Par exemple, on souhaite obtenir les affichages suivants :

```
>>>jeu_motus('MOTUS')
M----
donner votre proposition : MATIN
M-T--
donner votre proposition : MATHS
M-T-S
donner votre proposition : MOTOS
MOT-S
donner votre proposition : MOTUS
Félicitations, vous avez gagné en 4 coups.
```

Expliquer pourquoi le programme suivant ne donne pas toujours le résultat attendu.

```
def jeu_motus(mot):
    n=len(mot)
    nb_coups=0
    gagne=False
    print(mot[0]+'-'*(n-1))
    while nb_coups<n and gagne==False:
       mot_joueur=input('donner votre proposition : ')
       if mot_joueur==mot:
           gagne=True
       else:
           print(test_mot(mot, mot_joueur))
    if gagne:
       print('Félicitations, vous avez gagné en ', nb_coups, 'coups.')
    else:
       print('perdu, il fallait trouver : ',mot)
```

Exercice 3 (4 points)

1) Pour la réservation de billets à un concert, deux guichets utilisent une même variable `compteur` comptabilisant le nombre de places restantes. Celle-ci est stockée dans une mémoire partagée par les deux guichets. À chaque fois qu'un billet est délivré, la variable `compteur` diminue de 1.

Deux processus P1 et P2 associés à chacun des guichets accèdent en concurrence à la variable `compteur`.

Le déroulement de l'exécution des processus s'organise comme ci-dessous (du haut vers le bas). Des variables `compteur1` et `compteur2` sont spécifiques respectivement aux processus P1 et P2.

	Processus P1	**Processus P2**
Intervalle de temps 1	`compteur1 ← compteur`	

Intervalle de temps 2		`compteur2 ← compteur`
		`compteur2 ← compteur2 – 1`
		`compteur ← compteur2`

Intervalle de temps 3	`compteur1 ← compteur1 – 1` `compteur ← compteur1`	

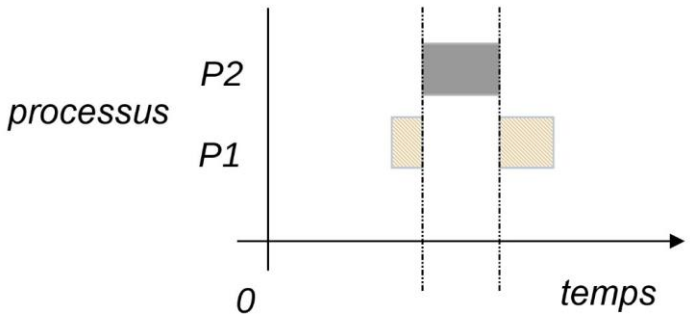


Figure 1

La valeur initiale de la variable `compteur` est 100 (en base 10).

a. **Montrer** que la valeur de la variable `compteur` après l'intervalle de temps 3 est égale à 99.
b. **Modifier** l'ordonnancement des processus pour, qu'à la fin, la valeur de la variable `compteur` soit égale à 98.

2) Une autre solution est envisagée.

La mémoire partagée est alors gérée par un jeton (appelé aussi sémaphore ou drapeau) de type exclusion mutuelle qui autorise ou non son accès. Ce jeton sera nommé `mutex`.

La classe `Jeton` contient les attributs et les méthodes suivants. Le jeton `mutex` est donc créé comme une instance de la classe `Jeton`.

Nom de la classe	Jeton
Attributs	id // identifiant état // jeton : libre ou occupé ...

Méthodes	verrouillage() // prise du jeton dès qu'il est libre
	déverrouillage() // libération du jeton
	...

L'algorithme des processus est donné ci-dessous (du haut vers le bas).

Processus P1	**Processus P2**

`mutex.verrouillage()`
`compteur1 ← compteur`
`compteur1 ← compteur1 – 1`
`compteur ← compteur1`
`mutex.deverrouillage()`
	`mutex.verrouillage()`
	`compteur2 ← compteur`
	`compteur2 ← compteur2 – 1`
	`compteur ← compteur2`
	`mutex.deverrouillage()`

a. **Justifier** que cette nouvelle solution permet à la variable `compteur` de comptabiliser correctement le nombre de places restantes.
b. **Proposer** un algorithme identique pour les processus P1 et P2 produisant le même résultat.

3) Pour l'organisation du concert, l'administrateur crée des guichets avec des priorités d'attribution. Quatre processus P1, P2, P3, P4 sont associés aux quatre guichets et exécutent chacun l'algorithme de la question 2 b. L'ordonnancement choisi est un ordonnancement avec priorité, sans un nouveau calcul de la priorité.

L'allocation du processeur par tranche (quantum) de temps est de 15 ms. Le quantum est insécable (c'est-à-dire qui ne peut pas être découpé).

Le détail des processus est donné ci-dessous avec des priorités allant de 0 définissant une priorité haute à 20 définissant une priorité basse.

processus	**priorité**	**Date arrivée sur le processeur**	**Durée service**
P1	5	30 ms	30 ms
P2	5	15 ms	30 ms
P3	8	15 ms	30 ms
P4	20	15 ms	30 ms

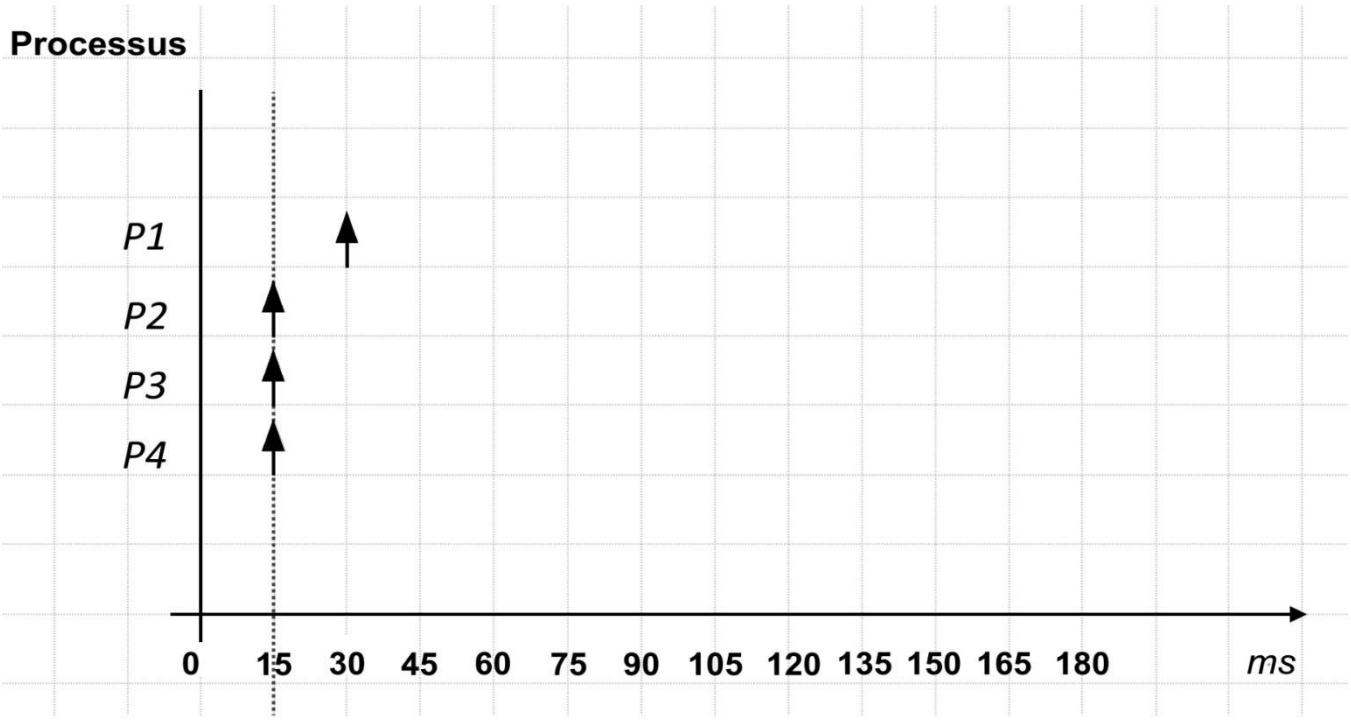
Compléter le diagramme **sur le document réponse**, à l'image de la figure 1 de la question 1.

DOCUMENT RÉPONSE À RENDRE OBLIGATOIREMENT ET DANS SON INTÉGRALITÉ AVEC LA COPIE

Exercice 3. Question 3)

Compléter le diagramme ci-dessous.

↑ indique la date d'arrivée des processus P1 à P4.



Modèle CCYC : ©DNE

NOM DE FAMILLE (naissance) :
(en majuscules)

PRENOM :
(en majuscules)

N° candidat :

N° d'inscription :

(Les numéros figurent sur la convocation, si besoin demander à un surveillant.)

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Né(e) le : ☐☐ / ☐☐ / ☐☐☐☐

1.2