CESI ÉCOLE D'INGÉNIEURS	**Concours commun CESI** **EPREUVE DE SCIENCES PHYSIQUES POUR L'INGENIEUR** **11 avril 2024**

À LIRE ATTENTIVEMENT AVANT DE COMMENCER

L'épreuve de Sciences Physiques pour l'Ingénieur dure 3 heures. L'énoncé comporte 4 problèmes. Chaque candidat doit en réaliser 3, en fonction de la spécialité choisie à l'inscription.

Les candidats inscrits à la spécialité « **Généraliste** » (code candidat commençant par « G ») réaliseront :

- Le problème I : Athlétisme et physique
- Le problème II : Moteur asynchrone
- Le problème IIIa : Ondes ultrasonores

Les candidats inscrits à la spécialité « **Informatique** » (code candidat commençant par « I ») réaliseront :

- Le problème I : Athlétisme et physique
- Le problème II : Moteur asynchrone
- Le problème IIIb : Informatique à l'hôpital

T.S.V.P.

Epreuve de physique et d'informatique

Durée : 3H00
Calculatrice non graphique autorisée

Consignes et avertissements :

- Ecrivez au **stylo à bille bleu ou noir** (PAS d'encre effaçable)
- N'utilisez pas de correcteur blanc, mais rayez proprement
- Ecrivez *lisiblement*, avec une taille de caractères raisonnable
- Numérotez vos copies
- Ne rendez pas l'énoncé
- Respectez la numérotation des questions
- Faites les problèmes et les questions des problèmes dans l'**ordre**
- Laissez des espaces pour les questions non traitées
- Encadrez vos résultats

Le respect rigoureux de ces consignes sera pris en compte (environ 3 % de la note finale).

Ce sujet comporte 3 problèmes indépendants, chacun de durée approximative 1h00 et comptant pour environ 30 % dans la notation finale.

Attention, le problème III est différent pour les candidats de la filière générale et ceux de la filière informatique.

Problème I - Athlétisme et physique

On peut analyser les performances obtenues en athlétisme grâce à la physique. En particulier, pour les courses de vitesse et les sauts, les muscles situés devant les cuisses, appelés quadriceps, jouent un rôle essentiel.

Tous les mouvements sont étudiés par rapport au référentiel terrestre galiléen. On fournit les valeurs numériques utiles :

- Accélération locale de la pesanteur : $g = 9,8$ m.s^{-2}
- Energie moyenne délivrée par les quadriceps d'une jambe : 500 J
- Masse moyenne d'un sprinteur ou d'un sauteur masculin : 80 kg
- Masse d'un marteau (lancer masculin) : 7,2 kg

1. Une formule obtenue en biomécanique indique que l'énergie cinétique d'une jambe est égale au sixième du produit de sa masse par le carré de la vitesse du coureur.

(a) Sachant que la masse d'une jambe représente 20 % de la masse corporelle, en déduire la valeur numérique de la vitesse du coureur, en m/s.

(b) Comparer avec la vitesse de pointe d'Usaïn Bolt à 43 km/h.

(c) Commenter.

Pour les deux questions suivantes, on suppose qu'un sauteur atteint la vitesse de 10 m/s juste avant le début de son saut.

2. On s'intéresse au saut à la perche. Calculer la hauteur maximale par rapport au sol que le centre de gravité d'un perchiste peut atteindre. On suppose que la perche a une masse négligeable et que le centre de gravité du perchiste est initialement à 1,0 m du sol.

3. On s'intéresse au saut en longueur. A l'aide d'une analyse dimensionnelle, en supposant que la longueur δ du saut ne dépend que de la vitesse V du sauteur et de l'accélération de la pesanteur g, établir une relation exprimant δ (à un facteur sans dimension près, pris égal à 1). Faire l'application numérique (A.N.).

On s'intéresse maintenant au lancer du poids, afin de déterminer l'angle de lancement optimal garantissant la plus grande portée. On suppose que le poids, assimilé à un point matériel de masse m_p, est lancé à une hauteur h_0 par rapport au sol, avec une vitesse initiale de norme v_0, faisant un angle α avec l'horizontale. On suppose également que le mouvement s'effectue sans frottement dans un plan orienté par un repère (O, x, z), l'axe x est horizontal, l'axe z est vertical ascendant et l'origine O est situé au sol, sur la verticale passant par le point de lancement du poids.

4. Etablir les équations différentielles vérifiées par les coordonnées $x(t)$ et $z(t)$ du poids. Préciser les conditions initiales $x(t = 0)$, $z(t = 0)$, $\dot{x}(t = 0)$ et $\dot{z}(t = 0)$.

5. Déterminer les expressions complètes de $x(t)$ et $z(t)$.

6. En déduire l'équation $z(x)$ de la trajectoire. Quelle est la forme géométrique de cette courbe ?

7. En déduire la portée L du lancer, en fonction de v_0, g, h_0 et α.

On s'intéresse maintenant au lancer du marteau. Ce projectile est constitué d'un poids, de masse m, auquel est attaché un câble métallique avec une poignée (cf figure 1). Cette étude prend en compte la force de frottement avec l'air.

FIGURE 1 – Lanceur de marteau débutant son lancer (source : AFP).

8. Dans la phase finale du lancer, le lanceur effectue 3 rotations par seconde sur lui-même. La distance entre l'axe de rotation et le centre de gravité du marteau est de 1,3 m. A quelle vitesse (en m/s) est lâché le marteau ?
9. On suppose que la force de frottement avec l'air s'écrit $-k\vec{v}$. Etablir l'équation vérifiée par le vecteur vitesse $\vec{v}$ du marteau. Quelle est la vitesse limite $\vec{v}_{lim}$ théoriquement atteinte.
10. On suppose désormais que la force de frottement est quadratique avec la vitesse (le coefficient de proportionnalité étant noté K). On résoud numériquement les équations du mouvement par la méthode d'Euler, à l'aide du programme suivant, écrit en Python 3. Ecrivez les 4 lignes où il manque du code :

```
from math import sin, cos, pi, sqrt

m = 7.2 # masse (kg)
K = 0.05 # coeff de frottement
g = 9.8 # accél de pesanteur (m2/s)
v0 = 30.0 # vitesse initiale (m/s)
h0 = 1.7 # hauteur initiale de lancer (m)
alpha_deg = 40.0 # angle de lancer (°)

dt = 1e-2 # pas de tps pour Euler (s)

t = [0.]
x = [0.]
z = [h0]
vx = [v0*cos(alpha_deg*pi/180.)]
vz = [v0*sin(alpha_deg*pi/180.)]

while z[-1] > 0. :
    v_norme = sqrt(vx[-1]**2 + vz[-1]**2)
    Fx = -K*v_norme * vx[-1]
    Fz = -K*v_norme * vz[-1]
    t.append(t[-1] + dt)
    vx.append(...) # ligne 1 à compléter
    vz.append(...) # ligne 2 à compléter
    x.append(...)  # ligne 3 à compléter
    z.append(...)  # ligne 4 à compléter
```

Problème II - Moteur asynchrone

Parmi les types de moteurs électriques, le moteur asynchrone se distingue par la simplicité de son rotor et son coût réduit. Il est notamment utilisé en traction ferroviaire (TGV Dasie, Eurostar MTST) et comme moteur de certaines voitures électriques (Tesla Roadster). On étudie ici le principe de fonctionnement de ce moteur et ses caractéristiques.

Le stator est la partie fixe du moteur. Son rôle est de créer un champ magnétique de norme constante, tournant à une vitesse angulaire contrôlable. Il est essentiellement constitué de bobines, judicieusement placées et orientées, parcourues par un courant alternatif sinusoïdal.

11. Une bobine du stator est modélisée par un solénoïde infini, d'axe Oz, de rayon R, ayant n spires par unité de longueur, parcouru par un courant I supposé constant (figure 2 à gauche). On admet que le champ magnétique est nul à l'extérieur du solénoïde. En se plaçant dans le système de coordonnées cylindriques (r, θ, z), étudier les symétries et invariances pour le champ magnétique $\vec{B}(M)$ en tout point M situé à l'intérieur du solénoïde.

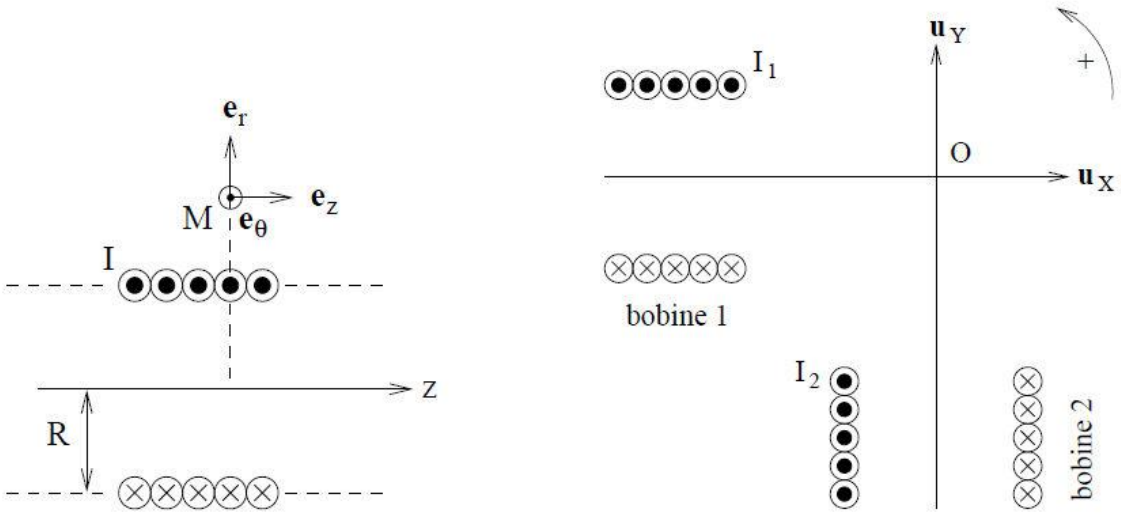

FIGURE 2 – A gauche : Solénoïde infini. A droite : Modélisation du stator.

12. Etablir que le champ magnétique à l'intérieur du solénoïde s'écrit :

$$\vec{B}(M) = \mu_0 n I \vec{e}_z$$

13. On admet qu'une bobine de longueur finie génère le long de son axe un champ magnétique ayant la même expression qu'à la question 12, y compris pour les points de l'axe situés à l'extérieur de la bobine. On suppose valide l'Approximation des Régimes Quasi-Stationnaires (ARQS). On dispose deux bobines identiques à angle droit (figure 2 à droite), alimentées par des courants alternatifs sinusoïdaux avec :

$$I_1(t) = I_0 \cos(\omega t) \text{ et } I_2(t) = I_0 \sin(\omega t)$$

Montrer que le champ magnétique total en O est un champ magnétique tournant, de norme B_0 à préciser, avec une vitesse angulaire Ω_B et un sens de rotation à déterminer. Indication : Travailler en coordonnées polaires.

Le rotor est la partie mobile du moteur. Dans un moteur asynchrone, il est constitué soit de bobinages reliés pour former un circuit électrique fermé, soit d'une structure cylindrique appelée cage d'écureuil. Dans cette étude, on modélise le rotor par une unique spire conductrice carrée de côté a, de résistance électrique R_e, capable de tourner sans frottement autour d'un axe OZ (cf figure 3) ; on notera Ω_m sa vitesse angulaire. L'action de la pesanteur est négligée. On applique dans tout l'espace un champ magnétique de norme constante, perpendiculaire à l'axe OZ, tournant avec une vitesse angulaire Ω_B constante, dans le sens +. A l'instant initial, le champ magnétique est selon $\vec{u}_X$ et le rotor est immobile dans le plan OXZ. On rappelle enfin que le couple exercé sur un dipôle magnétique, de moment dipôlaire $\vec{m}$, soumis à un champ magnétique $\vec{B}$, vaut $\vec{m} \wedge \vec{B}$.

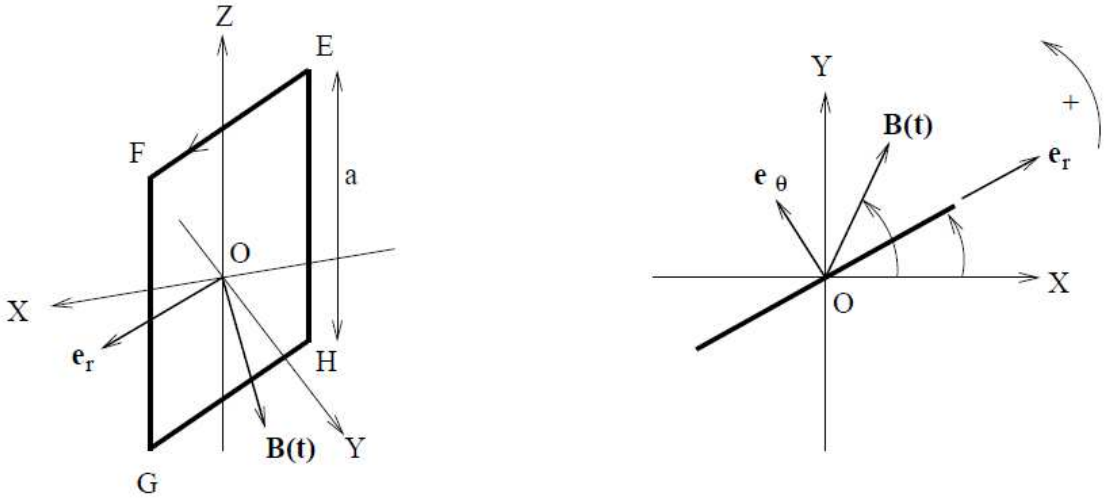

FIGURE 3 – A gauche : Vue en perspective du rotor. A droite : Vue de dessus du rotor.

14. Expliquer qualitativement, mais rigoureusement, ce qui va se passer.

15. Calculer l'intensité du courant qui circule dans la spire, en fonction de R_e, a, B_0, Ω_B et Ω_m. La convention d'orientation est précisée sur la figure 3 à gauche.

16. En déduire que le couple moteur moyen Γ_m par rapport à l'axe OZ, s'exerçant sur le rotor, vaut :

$$\Gamma_m = \frac{a^4 B_0^2}{2R_e}(\Omega_B - \Omega_m)$$

17. On suppose que le rotor est également soumis à un couple résistant (associé au système mécanique qu'il met en rotation) $-\Gamma_r$ (avec $\Gamma_r > 0$). Etablir la condition qui permet au moteur de démarrer.

18. Pour une valeur donnée de Γ_r, compatible avec un fonctionnement moteur, discuter de la stabilité du point de fonctionnement.

19. Discuter du cas $\Omega_B < \Omega_m$ et de son éventuel intérêt pratique.

20. En réalité, il faut aussi prendre en compte l'inductance propre du rotor. Le couple moteur varie alors en fonction de la vitesse angulaire Ω_m du rotor selon la courbe représentée sur la figure 4. Commenter (démarrage, modes de fonctionnement, stabilité des points de fonctionnement).

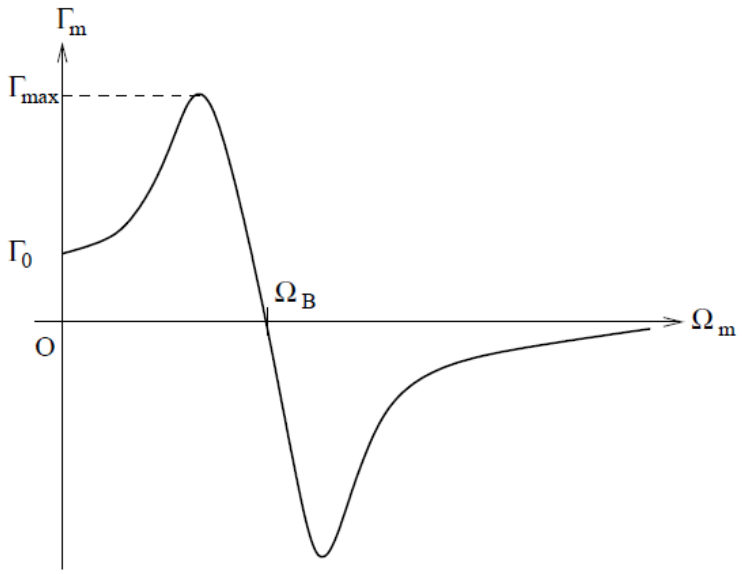

FIGURE 4 – Evolution du couple moteur en fonction de la vitesse angulaire du rotor.

Problème IIIb - Informatique à l'hôpital

ATTENTION, seuls les candidats de la filière informatique doivent traiter ce problème.

Tous les programmes et fonctions demandées doivent être écrits en **Python 3**. Il est inutile de commenter les programmes ou les fonctions que vous concevez. Sur les exemples présentés, le symbole `>>>` désigne l'invite de commande de l'interpréteur Python. Enfin, cet énoncé précise les annotations de type pour les fonctions à concevoir. Par exemple :

```
ma_fonction(a: int, LL: list, test: bool) -> dict
```

indique que le paramètre `a` est de type entier, `LL` est de type liste, `test` est de type booléen et que la fonction `ma_fonction` renvoie un dictionnaire. Cependant, sur votre copie, vous n'écrirez pas ces annotations, mais juste `ma_fonction(a, LL, test)`.

31. Un numéro de sécurité sociale (NSS) est formé de 15 chiffres décimaux : les 13 premiers donnent des informations sur le sexe, la date de naissance, ... alors que les 2 derniers chiffres constituent une clé de contrôle, permettant de confirmer la validité du NSS (et d'éviter notamment des erreurs de recopie). La clé de contrôle se calcule comme la soustraction entre 97 et le reste de la division entière (modulo) par 97 du nombre formé par les 13 premiers chiffres. Ecrire une fonction `est_valide(numero: str) -> bool`, qui prend comme paramètre une chaîne de caractères constitué de 15 chiffres décimaux et qui renvoie un booléen indiquant si le NSS est valide. On rappelle que l'opérateur *modulo* est le symbole `%` en Python et que la fonction `int()` permet de convertir une chaîne de caractères constituée de valeurs numériques en l'entier correspondant. On rappelle aussi le procédé de slicing, applicable sur n'importe quelle séquence. Exemples :

```
>>> 19%5  # reste de la division de 19 par 5
4
>>> int("321")
321
>>> est_valide("105016400132142")
False
>>> est_valide("212122725463922")
True
>>> "bonjour"[1:5:2]  # slicing
"oj"
```

32. La gestion des patients arrivant aux urgences est un problème important. Dans l'approche la plus simple, l'idée est que le premier patient qui arrive est traité en premier. Le type abstrait correspondant en informatique s'appelle une file. On va implémenter ce type à l'aide de listes Python. Par convention, le premier élément de la liste correspond au NSS du 1er patient arrivé (et qui sera donc pris en charge en premier). Ecrire 3 fonctions :

- `est_vide(file_d_attente: list) -> bool` qui renvoie un booléen indiquant si la file d'attente est vide ou pas,
- `nouvel_arrivant(file_d_attente: list, NSS: str) -> None` qui ajoute à la file d'attente le numéro de sécurité sociale d'un nouveau patient qui arrive aux urgences,
- `prise_en_charge(file_d_attente: list) -> str` qui retire de la file d'attente le NSS du patient arrivé en premier, et qui renvoie ce NSS.

On rappelle que la commande `pop(i)` appliqué à une liste `L`, écrit comme `L.pop(i)` retire l'élement d'indice `i` de la liste `L` et le renvoie.

33. La gestion précédente n'est pas satisfaisante car certains patients, dont la vie est en danger, devraient être prioritaires devant ceux qui ont des symptomes bénins. C'est pourquoi on va définir un score de priorité pour chaque patient arrivant, qui est un nombre entier d'autant plus élevé que l'exige l'état du patient et sa prise en charge rapide. On propose la formule suivante :

```
score = score_gravité + score_age
```

où `score_gravité` vaut 25 si l'état du patient est grave et que sa vie est en danger, 10 si l'état du patient est grave mais qu'il n'est pas immédiatement en danger de mort, 0 sinon, et où `score_age` vaut 5 si le patient a 6 ans ou moins ou bien 70 ans ou plus, et 0 sinon. De plus, pour tenir compte de l'attente qui peut se prolonger pour certains patients, les scores de tous les patients dans la file seront augmentés de 1, chaque fois qu'un nouveau patient arrive (celui-ci ne bénéficie pas de cette augmentation). La file d'attente est maintenant remplacée par une file dite de priorité, implémentée sous la forme d'une liste, où chaque élément est une liste de 2 éléments (un NSS et un score) représentant un patient ; cette liste est ordonnée selon les valeurs décroissantes de score des patients. Il n'y a pas de priorité particulière entre des patients qui ont le même score. Par exemple :

```
>>> file_de_priorité
[["129...", 31], ["122...", 18], ["287...", 18], ["131...", 5]]
```

Ecrire une fonction :

```
insere_nouveau_patient(NSS: str,
                       score_gravité: int,
                       age: int,
                       file_de_priorité: list) -> None
```

qui a pour but d'actualiser une file de priorité lorsqu'un nouveau patient arrive, caractérisé par son NSS, son score de gravité et son âge. On rappelle que la commande `insert(i, e)` appliqué à une liste `L`, écrit comme `L.insert(i, e)`, avec `i` compris entre `0` et `len(L)-1` insère dans la liste `L` l'élement `e` à la position d'indice `i` (ou l'ajoute si `i = len(L)`).

Un autre problème qui se pose est la répartition des patients dans les chambres d'un hôpital. On suppose que l'hôpital admet un rez-de-chaussée et 4 étages. Chaque niveau admet 10 chambres ; chaque chambre peut accueillir 0, 1 ou 2 patients. On modélise cette structure à l'aide d'une liste `H` de `N`=5 éléments, chaque élément étant lui-même une liste de `P`=10 entiers (dont les valeurs sont 0, 1 ou 2). Par convention, `H[0]` désigne le rez-de-chaussée et `H[1][0]` indique le nombre de patients dans la chambre n°0 de l'étage n°1. Les variables `N` et `P` sont des variables globales.

34. Ecrire un programme Python qui créé la liste `H`, représentant l'hôpital avec aucun patient dedans.

35. On suppose qu'il y a `Q` patients (avec `Q` compris entre 0 et `NxPx2`) et on veut les placer, dans un hôpital initialement vide de patients, de manière progressive, en remplissant intégralement les étages inférieurs en priorité. De plus, chaque étage doit être remplis de façon à ce que les chambres de plus petit numéro soient complètement occupées. Ecrire une fonction Python :

```
repartit_les_patients(H: list, Q: int) -> None
```

qui modifie la liste `H` pour effectuer cette répartition. Exemple :

```
>>> repartit_les_patients(H, 67)
>>> H
[[2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
 [2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
 [2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
 [2, 2, 2, 1, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

36. On préfère séparer le plus possible les patients pour limiter les infections. Dans un hôpital toujours initialement vide de patients, l'idée est de répartir les patients si possible 1 par chambre, dans des étages si possibles différents. On ne se préoccupe pas d'autres distanciations et on continue de remplir les chambres d'un étage selon l'ordre croissant de leur numéro et les étages selon l'ordre croissant de leur numéro. Ecrire une fonction Python :

```
repartit_mieux_les_patients(H: list, Q: int) -> None
```

qui modifie la liste `H` pour effectuer cette répartition. Exemple :

```
>>> repartit_mieux_les_patients(H, 67)
>>> H
[[2, 2, 2, 2, 1, 1, 1, 1, 1, 1],
 [2, 2, 2, 2, 1, 1, 1, 1, 1, 1],
 [2, 2, 2, 1, 1, 1, 1, 1, 1, 1],
 [2, 2, 2, 1, 1, 1, 1, 1, 1, 1],
 [2, 2, 2, 1, 1, 1, 1, 1, 1, 1]]
```

On s'intéresse enfin à une base de données relationnelle de l'hôpital, rassemblant des informations sur les patients et les médecins. La base comprend 4 tables, dont les schémas relationnels sont :

- PATIENT : NSS (int), NomPatient (str), Age (int)
- SERVICE : IdService (int), Denomination (str), Etage (int)
- MEDECIN : IdMedecin, NomMedecin (str), IdServ ♯ (int)
- PRISEENCHARGE : IdPEC (int), NSSPat ♯ (int), IdMed ♯ (int)

Les attributs soulignés désignent des clés primaires et les attributs suivis du symbole ♯ désignent des clés étrangères (dont les noms sont suffisamment implicites pour éviter toute ambiguïté). Noter que le NSS d'un patient est désormais défini comme un entier (et non pas comme une chaîne de caractères). Ci-après sont affichées les 4 premières lignes de chaque table.

PATIENT		
NSS	NomPatient	Age
1234	Dupont	30
2251	Albert	74
1458	Joe	60
2111	Farah	23

SERVICE		
IdService	Denomination	Etage
1	Neurologie	2
2	Chirurgie	2
3	Ophtalmologie	1
4	Urgences	0

MEDECIN		
IdMedecin	NomMedecin	IdServ
1	Starck	1
2	Bernard	1
3	Abdoul	2
4	Bernstein	4

PRISEENCHARGE		
IdPEC	NSSPat	IdMed
1	1234	1
2	2251	4
3	1458	3
4	2111	3

On indique ci-après les commandes SQL autorisées (à l'exclusion de toute autre) : SELECT, DISTINCT, WHERE, JOIN ... ON ..., ORDER BY, DESC, GROUP BY, HAVING et les fonctions d'agrégation : MIN(), MAX() et COUNT().

37. Ecrire une requête SQL qui affiche le nom des patients, écrit dans l'ordre croissant de leur âge.

38. Ecrire une requête SQL qui affiche le nom des médecins travaillant à l'étage 2.

39. Ecrire une requête qui indique l'étage dans lequel est situé la chambre du patient se nommant "Dupont"

40. Ecrire une requête SQL qui affiche le nom et la spécialité du (ou des) médecin qui prend en charge au moins 2 patients.