LE GOUVERNEMENT DU GRAND-DUCHÉ DE LUXEMBOURG Ministère de l'Éducation nationale, de l'Enfance et de la Jeunesse	**EXAMEN DE FIN D'ÉTUDES SECONDAIRES CLASSIQUES** **2019**		
BRANCHE	**SECTION(S)**	**ÉPREUVE ÉCRITE**	
Informatique	**B**	*Durée de l'épreuve :*	*180 minutes*
		Date de l'épreuve :	

Ajouter le nom du lycée et le numéro personnel sous forme de commentaire tout au début de chaque code-source ! Enregistrer les deux fichiers `.py` demandés dans votre répertoire de travail.

Les seules importations permises sont celles de `pygame`, `pygame.locals`, `sys` et `random`.

Exercice 1 : Une classe de matrices (30 points)

Écrire un programme `matrices.py` qui implémente une classe `Matrix` représentant une matrice carrée de nombres réels :

1. La classe `Matrix` a deux attributs : `n` est l'ordre de la matrice (le nombre de lignes et de colonnes) ; `m` est une liste de listes, contenant les éléments de la matrice. Le constructeur reçoit comme argument obligatoire l'ordre n de la matrice à créer. Il initialise l'attribut `n` de la matrice et crée l'attribut `m` dont les n^2 éléments sont tous nuls. **[2 p.]**

2. La méthode `copy` crée et renvoie une copie de la matrice de départ (également de type `Matrix`). Les éléments de la copie doivent pouvoir être modifiés indépendamment des éléments de la matrice de départ ! **[2 p.]**

3. La méthode `random_elements` reçoit comme argument obligatoire un nombre naturel $x > 0$; elle remplace tous les éléments de la matrice par des nombres entiers aléatoires compris entre $-x$ et x (bornes comprises). **[2 p.]**

4. La méthode `identity` remplace les éléments de la matrice par ceux de la matrice identité E.
   Rappel : $e_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$ **[2 p.]**

5. La méthode `plus` reçoit comme argument obligatoire une 2$^{\text{e}}$ matrice (`self` étant la 1$^{\text{re}}$ matrice) ; elle calcule et renvoie une nouvelle matrice qui est la somme des deux matrices données. **[3 p.]**

6. La méthode `times` reçoit comme argument obligatoire une 2$^{\text{e}}$ matrice ; elle calcule et renvoie une nouvelle matrice qui est le produit des deux matrices données. **[3 p.]**
   Rappel : si $C = A \cdot B$ (Python : `c = a.times(b)`), alors $c_{ij} = \sum_{1 \leq k \leq n} a_{ik} b_{kj}$.

7. La méthode `power` reçoit comme argument obligatoire un nombre naturel p ; elle calcule et renvoie une nouvelle matrice qui est la $p^{\text{ième}}$ puissance de la matrice `self`.
   Rappel : $A^p = A \cdots A$ (avec p facteurs A dans le membre de droite). **[3 p.]**
   Cas particuliers : $A^0 = E$ (matrice identité), et $A^1 = A$.

8. La méthode `maxabs` renvoie le maximum des valeurs absolues de tous les éléments de la matrice. **[3 p.]**

9. La méthode `exp` reçoit comme argument facultatif un nombre `epsilon` (valeur par défaut : 0,000 001). Elle calcule et renvoie une nouvelle matrice qui est une approximation de l'*exponentielle* de la matrice donnée : on admet que

$$\exp A \overset{\text{déf}}{=} \lim_{N\to+\infty} \sum_{k=0}^{N} \frac{A^k}{k!} \approx \boxed{E + A + \frac{A^2}{2} + \frac{A^3}{6} + \frac{A^4}{24} + \cdots + \frac{A^N}{N!}} \quad \text{avec } N \text{ suffisamment grand.}$$

   La méthode `exp` implémente la formule encadrée.

   Rappel : $k! = k \cdot (k-1) \cdots 2 \cdot 1$. La fraction $\frac{A^k}{k!}$ signifie que tous les éléments de A^k doivent être divisés par le nombre $k!$.

   Choix de N : dès que `maxabs` appliquée à la matrice $\frac{A^k}{k!}$ donne une valeur strictement inférieure à ε, on arrête les calculs après avoir ajouté ce dernier terme à la somme. **[5 p.]**

10. La méthode `display` affiche la matrice à l'écran, de manière semblable à l'exemple d'exécution. **[2 p.]**

Le programme principal demande à l'utilisateur d'entrer l'ordre n de la matrice (on peut supposer qu'il fournit une valeur entre 2 et 10), crée une matrice carrée d'ordre n, remplit cette matrice avec des nombres aléatoires compris entre -9 et 9, l'affiche, calcule son exponentielle et affiche également cette nouvelle matrice. Voir l'exemple d'exécution. **[3 p.]**

Remarque : Le programme **n'a pas besoin** de traiter spécialement des entrées erronées (p. ex. mauvaise valeur entrée pour n) ou des appels de méthodes avec données incompatibles (p. ex. appel des méthodes `plus` ou `times` avec deux matrices d'ordres différents). De tels cas peuvent être négligés.

Exemple d'exécution : (l'utilisateur entre la valeur `3` à la 1re ligne)

```
Entrez n : 3
Matrice aléatoire A :
[[5, 1, 3]
 [9, -2, 9]
 [2, -6, -8]]
Matrice exp A :
[[181.92975262053199, -6.037040064990783, 37.476274303416524]
 [168.01621740863854, -5.572197517873239, 34.606470612505184]
 [-49.132343128907415, 1.634528257613829, -10.114046426747043]]
```

Exercice 2 : Le problème des huit dames (30 points)

Le but du *problème des huit dames* est de placer huit dames d'un jeu d'échecs sur un échiquier de 8×8 cases sans que des dames ne puissent se capturer mutuellement. Par conséquent, deux dames ne doivent jamais occuper une même ligne, colonne ou diagonale.

Durant le jeu, l'utilisateur pourra placer ou enlever des dames sur l'échiquier, et le programme le guidera en affichant des messages appropriés dans l'en-tête de la fenêtre `pygame`.

L'écran `pygame` a comme dimensions 420×420 pixels. En son centre, un échiquier de 400×400 pixels est représenté, dont chacune des 8×8 cases mesure donc 50×50 pixels. L'échiquier est entouré d'un cadre (carré) noir d'épaisseur 1 pixel. L'arrière-fond extérieur à l'échiquier est blanc.

Écrire un programme `queens8.py`, composé comme suit :

1. La fonction `draw_disc` reçoit comme arguments l'écran `pygame`, le numéro de colonne et le numéro de ligne d'une case de l'échiquier. Elle dessine un disque vert de diamètre 30 pixels, centré dans la case indiquée. **[2 p.]**

2. La fonction `draw_queens` reçoit comme arguments l'écran `pygame` et une liste de coordonnées de dames. Elle dessine toutes ces dames en appelant la fonction précédente. **[2 p.]**

3. La fonction `draw_board` reçoit comme argument l'écran `pygame`. Elle dessine un échiquier vide, semblable à la 1re capture d'écran, avec 32 cases noires et autant de cases blanches. La case en haut à gauche est obligatoirement blanche. **[4 p.]**

4. La fonction `draw_line` reçoit comme arguments l'écran `pygame` et les coordonnées de deux dames distinctes. Elle dessine un segment rouge d'épaisseur 6 pixels, reliant les centres des deux dames en question. **[2 p.]**

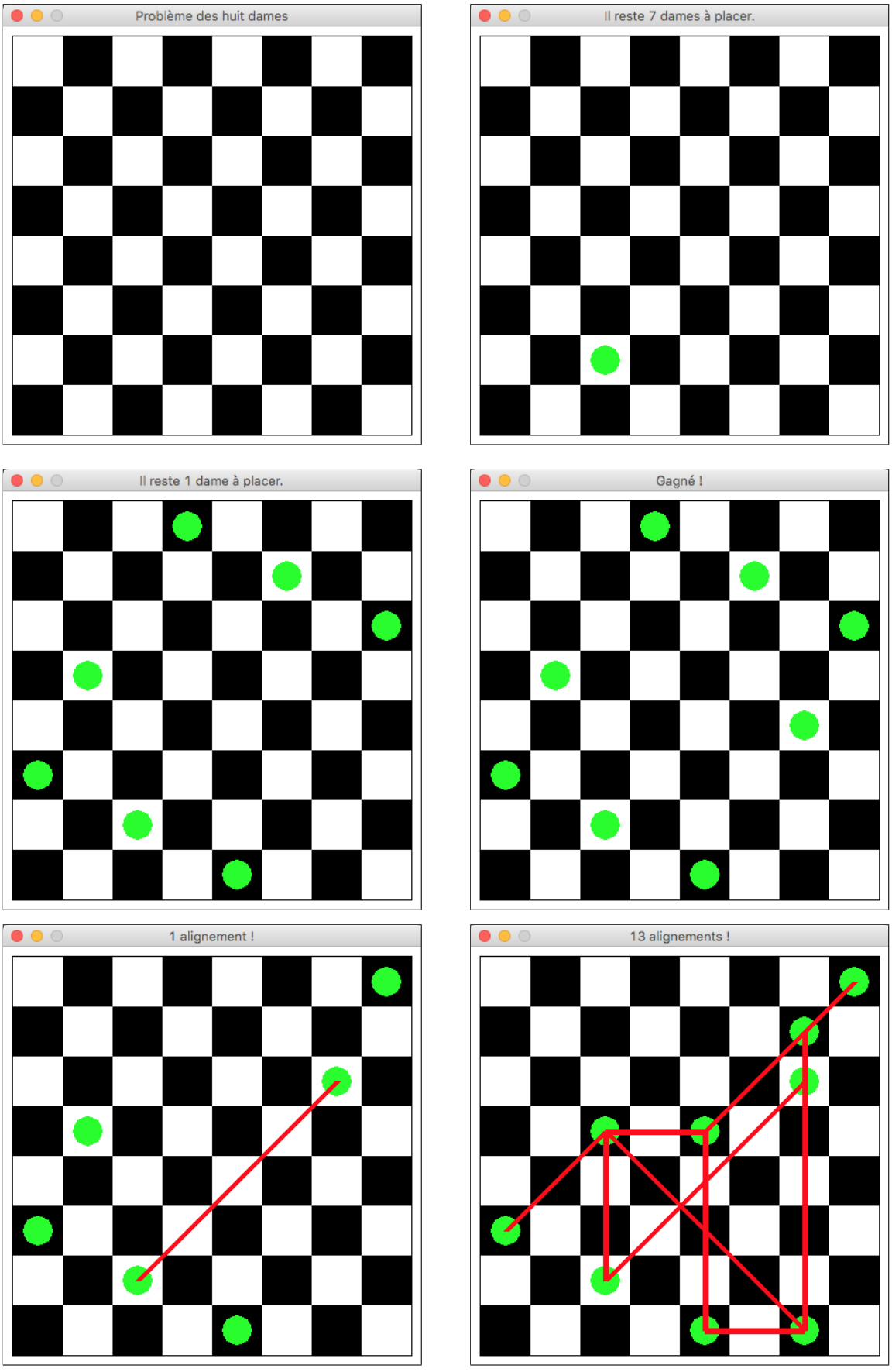
5. La fonction `find_alignments` reçoit comme argument une liste de coordonnées de dames. Elle renvoie une nouvelle liste contenant les coordonnées des paires de dames alignées. Chaque élément de cette liste est donc composé de 4 nombres (les coordonnées de deux dames). On dit que deux dames sont alignées ssi elles partagent la même ligne, colonne ou diagonale. **[5 p.]**

6. La fonction `show_alignments` reçoit comme arguments l'écran `pygame` et une liste de coordonnées de dames alignées (générée p. ex. par la fonction précédente). Elle dessine tous les segments rouges représentant les alignements, en appelant la fonction `draw_line`. **[2 p.]**

7. Boucle principale (pouvant être quittée par l'événement `QUIT`) : Si l'utilisateur clique dans une case vide de l'échiquier (bouton gauche de la souris, événement `MOUSEBUTTONDOWN`), une nouvelle dame y est placée. S'il clique dans une case occupée par une dame, celle-ci est enlevée. L'écran est ensuite mis à jour, c'est-à-dire toutes les dames actuellement présentes et tous les alignements sont représentés à l'aide des fonctions susmentionnées. L'échiquier peut comprendre entre 0 et 64 dames, et le nombre d'alignements peut ainsi varier entre 0 et 728. **[7 p.]**

   L'en-tête de la fenêtre contient à tout moment l'un des messages suivants, cf. captures d'écran :

   - ⋆ « Problème des huit dames » (uniquement lors du démarrage du programme),
   - ⋆ « 1 alignement ! » ou « n alignements ! » (avec $n \geq 2$) en cas d'alignements présents,
   - ⋆ « Gagné ! » lorsque 8 dames sont placées sans aucun alignement, sinon
   - ⋆ « Il reste 1 dame à placer. » ou « Il reste n dames à placer. » (avec $2 \leq n \leq 8$) **[6 p.]**


Problème des huit dames
Il reste 7 dames à placer.
Il reste 1 dame à placer.
Gagné !
1 alignement !
13 alignements !