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 CORRIGÉ – BARÈME		
BRANCHE	**SECTION(S)**	**ÉPREUVE ÉCRITE**	
Informatique	**B**	*Durée de l'épreuve :*	*180 minutes*
		Date de l'épreuve :	

1 Programme `matrices.py`

```
1  from random import randrange
2
3  class Matrix:
4      def __init__(self, n): # C: 2 p.
5          self.n = n
6          self.m = [x[:] for x in [[0] * n] * n]
7      def copy(self): # C: 2 p. (1 p. au maximum si la copie n'est pas indépendante de l'original)
8          m2 = Matrix(self.n)
9          m2.m = [x[:] for x in self.m]
10         return m2
11     def random_elements(self, x): # C: 2 p. (1 p. au maximum en cas d'erreur sur les bornes à inclure)
12         for i in range(self.n):
13             for j in range(self.n):
14                 self.m[i][j] = randrange(-x, x + 1)
15     def identity(self): # C: 2 p.
16         for i in range(self.n):
17             for j in range(self.n):
18                 if i == j: # plus compact : self.m[i][j] = 1 if i == j else 0
19                     self.m[i][j] = 1
20                 else:
21                     self.m[i][j] = 0
22     def plus(self, m2): # C: 3 p. (2 p. au maximum si self et/ou m2 sont modifiés par la méthode)
23         s = self.copy()
24         for i in range(s.n):
25             for j in range(s.n):
26                 s.m[i][j] += m2.m[i][j]
27         return s
28     def times(self, m2): # C: 3 p. (2 p. au maximum si self et/ou m2 sont modifiés par la méthode)
29         p = Matrix(self.n)
30         for i in range(self.n):
31             for j in range(self.n):
32                 for k in range(self.n):
33                     p.m[i][j] += self.m[i][k] * m2.m[k][j]
34         return p
35     def power(self, p): # C: 3 p. (2 p. au maximum si self est modifié par la méthode)
36         if p < 1: # pénalisation d'1 p. si les cas p=0 et/ou p=1 donnent un mauvais résultat
```

```
37              r = Matrix(self.n)
38              r.identity()
39          else:
40              r = self.copy()
41              for i in range(1, p):
42                  r = r.times(self)
43          return r
44      def maxabs(self): # C: 3 p. (2 p. au maximum si self est modifié par la méthode)
45          x = 0
46          for i in range(self.n):
47              for j in range(self.n):
48                  x = max(x, abs(self.m[i][j]))
49          return x
50      def exp(self, epsilon = 0.000001): # C: 5 p. (4 p. au maximum si self est modifié par la méthode)
51          e = Matrix(self.n)
52          e.identity()
53          e = e.plus(self) # 1 p. pour l'initialisation de la somme partielle
54          p = self.copy()
55          k = 1
56          f = 1
57          while k < 100: # 2 p. pour le mécanisme de la boucle (hors calcul de k!)
58              k += 1
59              p = p.times(self)
60              for i in range(p.n):
61                  for j in range(p.n):
62                      p.m[i][j] /= k # 1 p. pour la bonne valeur du dénominateur
63                                     # si l'élève écrit une fonction factorielle auxiliaire correcte => 1 p.
64              e = e.plus(p)
65              if p.maxabs() < epsilon: # 1 p. pour la condition et le mécanisme d'arrêt
66                  break
67          return e
68      def display(self): # C: 2 p. (1 p. au maximum si le formatage est mauvais, p.ex. crochets/virgules
            manquants)
69          print("[", end = "")
70          print(self.m[0])
71          for i in range(1, self.n - 1):
72              print("", self.m[i])
73          print("", self.m[-1], end = "") # ceci ne fonctionne pas si n = 1 (pas requis par l'énoncé)
74          print("]")
75
76  n = int(input("Entrez n : ")) # C: 3 p.
77  a = Matrix(n)
78  a.random_elements(9)
79  print("Matrice aléatoire A :")
80  a.display()
81  b = a.exp()
82  print("Matrice exp A :")
83  b.display()
```

2 Programme queens8.py

```
1  import pygame, sys
2  from pygame.locals import *
3
4  def draw_disc(s, x, y): # C: 2 p.
5      pygame.draw.circle(s, Color("green"), (35+50*x, 35+50*y), 15, 0)
6
7  def draw_queens(s, q): # C: 2 p.
8      for (x, y) in q:
9          draw_disc(s, x, y)
10
11 def draw_board(s): # C: 4 p.
12     s.fill(Color("white")) # 1 p.
13     pygame.draw.rect(s, Color("black"), (9, 9, 402, 402), 1) # 1 p. (si l'emplacement est exact)
14     for x in range(8):
15         for y in range(8):
16             if (x + y) % 2 == 1: # 1 p. si l'alternance des couleurs est correcte
17                 pygame.draw.rect(s, Color("black"), (10+50*x, 10+50*y, 50, 50), 0) # 1 p. pour le
                       dessin d'un carré
18
19 def draw_line(s, x1, y1, x2, y2): # C: 2 p.
20     pygame.draw.line(s, Color("red"), (35+50*x1, 35+50*y1), (35+50*x2, 35+50*y2), 6)
21
22 def find_alignments(q): # C: 5 p.
23     li = []
24     for i in range(len(q)):
25         for j in range(i + 1, len(q)): # 1 p. pour le mécanisme des boucles
26             (x1, y1) = q[i]
27             (x2, y2) = q[j]
28             if x1 == x2 or y1 == y2 or abs(x1 - x2) == abs(y1 - y2): # 3 p. pour les bonnes
                       conditions
29                 li.append((x1, y1, x2, y2)) # 1 p. pour la construction des éléments de la liste-résultat
30     return li
31
32 def show_alignments(s, c): # C: 2 p.
33     for (x1, y1, x2, y2) in c:
34         draw_line(s, x1, y1, x2, y2)
35
36 queens = [] # C: 13 p. (dont 6 pour les 6 en-têtes différentes)
37 pygame.init()
38 size = (420, 420)
39 screen = pygame.display.set_mode(size)
40 draw_board(screen) # 1 p. pour les initialisations
41 pygame.display.set_caption("Problème␣des␣huit␣dames") # en-tête: 1 p.
42
43 game_over = False
44 while not game_over: # 1 p. pour le mécanisme de la boucle principale (y compris la fin du programme)
45     pygame.display.update()
46     for event in pygame.event.get():
47         if event.type == QUIT:
48             game_over = True # 1 p. pour le mécanisme de sortie
49         if event.type == MOUSEBUTTONDOWN: # 2 p. pour l'étude des coordonnées de la souris
```

```
50              (mx, my) = pygame.mouse.get_pos()
51              x = (mx - 10) // 50
52              y = (my - 10) // 50
53              if 0 <= x < 8 and 0 <= y < 8:
54                  if (x, y) in queens: # 2 p. pour placer/enlever une dame et réafficher l'échiquier
55                      queens.remove((x, y))
56                  else:
57                      queens.append((x, y))
58                  draw_board(screen) # méthode brute : on redessine tout (nos ordinateurs sont
                        suffisamment rapides)
59                  draw_queens(screen, queens)
60                  align_list = find_alignments(queens)
61                  show_alignments(screen, align_list)
62                  if len(align_list) == 1:
63                      message = "1 alignement !" # en-tête: 1 p.
64                  elif len(align_list) > 1:
65                      message = str(len(align_list)) + " alignements !" # en-tête: 1 p.
66                  elif len(queens) == 8:
67                      message = "Gagné !" # en-tête: 1 p.
68                  elif len(queens) == 7:
69                      message = "Il reste 1 dame à placer." # en-tête: 1 p.
70                  else:
71                      message = "Il reste " + str(8-len(queens)) + " dames à placer." # en-tête: 1 p.
72                  pygame.display.set_caption(message)
73
74  pygame.quit()
75  sys.exit()
```