**Exercice : (4 points)**

***Approximation du nombre π***

La constante **Pi (π)** est l'un des nombres les plus importants et les plus fascinants dans le monde des mathématiques. C'est un nombre irrationnel, son écriture est faite d'une infinité de décimales, sans aucune périodicité. On n'arrive qu'à déterminer une écriture décimale approchée de sa valeur réelle. Pour cela, plusieurs méthodes pratiques et efficaces ont été employées.

La formule suivante a été publiée par le mathématicien indien **Nilakantha Somayaji** au 15e siècle, elle donne **11** décimales correctes de **π** :

$$\pi \approx 3 + \sum_{k=1}^{n} \frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$$

**Q1-** Écrire la fonction ***puiss (k)*** qui reçoit en paramètre un entier strictement positif **k**, et qui retourne la valeur de $(-1)^k$, en utilisant la parité de **k** : Si **k** est pair, alors la fonction retourne **1**. Et si **k** est impair, alors la fonction retourne **-1**.

préconditions ≡ hypothèses

puiss (-1) ("imad")

* (Q 1)

$$(-1)^k = \begin{cases} 1 & \text{si } k \text{ est pair} \\ -1 & \text{sinon (si } k \text{ est impair)} \end{cases}$$

```
def puiss(k):
    if k%2 == 0:
        return 1
    else:
        return -1
```

"7" → 7

sth → int

```
k= int(input('donner un entier positif'))
if k < 0 :
    print('donner un entier positif')
def puiss(k) :
    if k % 2 == 0 :
        return 1
    else :
        return -1
```

sth

fonctions → def (k)

exécution → k = ?

```
def puiss(k):
    if k%2==0:
        return 1
    else:
        return -1
```

k = 2 → 1

k = 3 → -1

```
def puiss(k):
    if k%2==0:
        return 1
    return -1
```

k = 2 → 1

k = 3 → -1

f

k%2 == 0 → 1

k%2 == 1 → -1

$$f(x) = 2(1-x) - 1$$

$f(0) = 1$ $\quad$ $f(1) = -1$

$f(k\%2)$

```
def puis(k):
    return 2*(1-(k%4))-1
```

0 → 0

$f$ : 1 → −1

$f(x) = -x$

0 → −1

$f$ : 1 → 0

$f(x) = 1 - (x\%2)$

---

```
def g(a,...):
    if a%2==0:
        return True
    else:
        return False
```

Astuce

(1) → False

(2)

```
def g(a,...):
    return a%2==0
```

True / False

**Q2-** Déterminer la complexité de la fonction ***puiss (k)***, avec justification.

Q2)

```
def puiss (k):
    if k%2 == 0:
        return 1
    return -1
```

2 opérations

?

$C(k) = O(1)$ ⟹ Complexité Constante

**Q3-** Écrire la fonction ***terme (k)*** qui reçoit en paramètre un entier strictement positif **k**, et qui retourne la valeur du terme suivant :

$$\frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$$

```
def terme(k):
    num = 4*puiss(k+1)
    den = 2*k*(2*k+1)*(2*k+2)
    return num/den
```

0.2

$\frac{a}{b \times c}$

$a/(b*c)$ ?

$\frac{a}{b} * c$ → 0.8

```
def terme(k):
    return 4*puiss(k+1)/(2*k+1)(2*k+2)
```

**Q4-** Écrire la fonction ***liste_termes (n)*** qui reçoit en paramètre un entier strictement positif **n**. La fonction retourne une nouvelle liste ***L*** qui contient les valeurs de ***terme(k)*** pour ***k=1, 2, 3, ... n***

* les listes en compréhension :

[f(a) for a in X]

[f(a) for a in X if C(a)]

* boucle for + append

```
def liste_termes(n):
    L = []
    for k in range(1, n+1):
        L.append(terme(k))
    return L
```

def liste_termes(n):
    return [terme(k) for k in range(1,n+1)]

range(a,b) → [a, a+1, ..., b-1]

extend : [ ]+[ ]

b = [5,9]  a = [0,1,2]

b.extend(a) → [5,9,0,1,2]

a = [.....]

x

O(1)

a.append(x)

a.extend([x])

n → len(a)

≃ O(n)

a = a + [x]

a += [x]

**Q5-** Écrire la fonction ***somme (L)*** qui reçoit en paramètre une liste **L** de nombres réels, et qui retourne la valeur de la somme des éléments de **L**, sans utiliser la fonction prédéfinie sum().

1ère Méthode:

sum(L)

```
def somme(L):
    s = 0
    for i in range(len(L)):
        s += L[i]
    return s
```

les indices de L

s += L[i] ⟺ s = s + L[i]

range(0,len(L)) ⟺ l'ensemble des indices de L

[0,3]

[0, 7, 5, 2]
0 1 2 3
-4 -3 -2 -1

---

s = 0 ⟺ on affecte à s la valeur 0 (s devient 0)

s == 0 ⟺ est-ce que s = 0 ?

2ème Méthode

```
def somme(L):
    s = 0
    for i in L:
        s += i
    return s
```

# les algos classiques :

* Somme
* max / min
* Comptage.

---

**Q6-** Écrire la fonction ***pi_nilakantha (n)*** qui reçoit en paramètre un entier strictement positif **n**, et qui retourne la valeur approchée de **π**, en utilisant la formule de Nilakantha.



$$\pi \approx 3 + \sum_{k=1}^{n} \frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$$

puis(k) → $(-1)^k$

terme(k) → $\frac{4(-1)^{k+1}}{2k(2k+1)(2k+2)}$

liste_terme(n) → [terme(k)] pour k = 1 ... n

Somme

```
def pi_nilakantha(n):
    termes = liste_termes(n)
    som_termes = somme(termes)
    return 3 + som_termes
```

```
def pi_nilakantha(n):
    pi =3
    L = liste_termes(n)
    pi += somme(L)
    return pi
```

✓

pi = 3

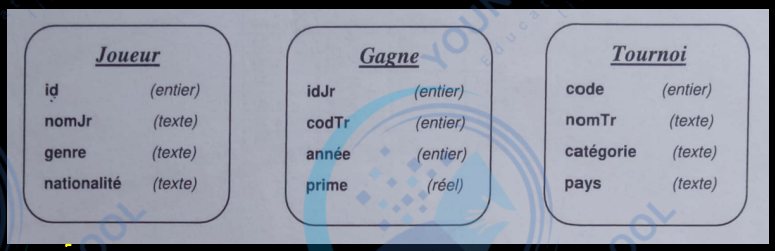
pi += Σ ... (k=)

return Pi

Pi = 3 + somme(L)

Σ

# * Partie I : SQL :




Joueur

| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

Tournoi

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

Gagne

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Q1)

**Q.1** – Rédiger une requête SQL qui supprime, dans la table **'Tournoi'**, tous les tournois des deux catégories **'WTA Premier Mandatory'** et **'WTA Premier 5'**.

ligne

du point de vue de la ligne

```
DELETE FROM Tournoi
WHERE catégorie = 'WTA Premier Mandatory'
OR catégorie = 'WTA Premier 5' ;
```

AND

à ne pas oublier

⚠ On ne supprime que des lignes !

```
DELETE FROM Tournoi WHERE nomTr LIKE 'WTA Premier Mandatory' AND 'WTA Premier 5' ;
```

OK

catégorie

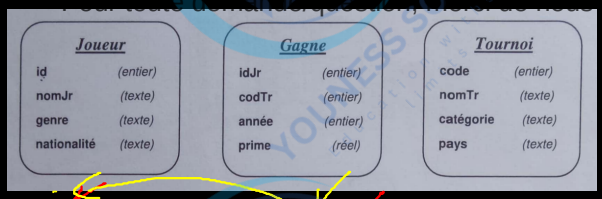
OR catégorie = ...

'imad %'

WHERE A = X OR A = Y

⇔

WHERE A IN ( X , Y )




| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

→ Joueur

Tournoi

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Gagne

**Q.2** – Rédiger une requête SQL, qui permet d'obtenir les différentes catégories des tournois, triées dans l'ordre alphabétique.

SELECT DISTINCT

ORDER BY

A → B → ... → Z

```
SELECT DISTINCT catégorie
FROM Tournoi
ORDER BY catégorie ASC;
```

DISTINCT

```
SELECT Catégorie FROM Tournoi ORDER BY Catégorie;
Avec des c minuscule pour catégorie *
```

ORDER BY
Col Mode
ASC
DESC
Croissant
descendant
par défaut

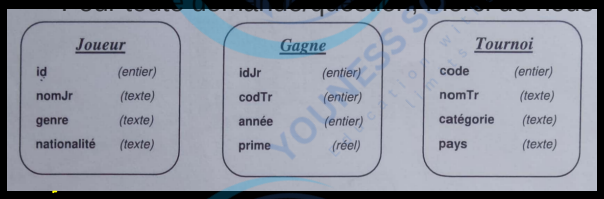
DISTINCT

```
SELECT catégorie FROM Tournoi where catégorie ORDER BY catégorie (asc) ;
```

filtre

Rep 1
Rep j

Interpreteur




| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

→ Joueur

Tournoi

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Gagne

**Q.3** – Rédiger en algèbre relationnelle, une requête qui permet d'obtenir les pays qui organisent des tournois dans la catégorie '**ATP 500**' et dans la catégorie '**Master 1000**'.

**Q.4** – Écrire la requête **Q.3** en langage SQL.

Q3)

SELECT ⟷ $\Pi$ (projection)

WHERE ⟷ $\sigma$ (sélection)

-cat → 'ATP 500'

→ 'Master 1000'

cat = X or cat = Y

(Jointure)

① $\Pi_{pays}(\sigma_{categorie = 'ATP 500' \vee Categorie = 'Master 1000'}(Tournoi))$

②

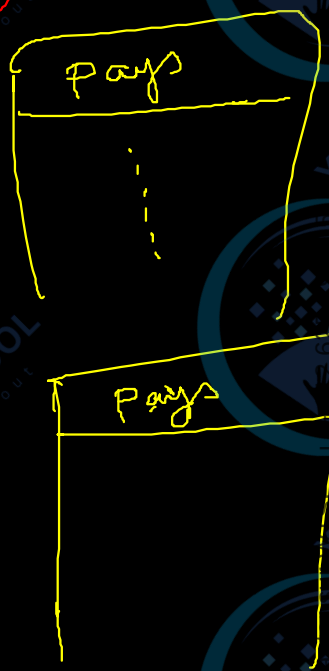

$\Pi_{pays}(\sigma_{categorie = 'ATP 500'}(Tournoi)) \cap \Pi_{pays}(\sigma_{categorie = 'Master 1000'}(Tournoi))$

(Q4) (I)

```
SELECT pays
FROM Tournoi
WHERE categorie = 'ATP 500'
OR Categorie = 'Master 1000' ;
```

(II)

```
SELECT pays
FROM Tournoi
WHERE categorie = 'ATP 500'
INTERSECT
SELECT pays
FROM Tournoi
WHERE categorie = 'Masters 1000' ;
```

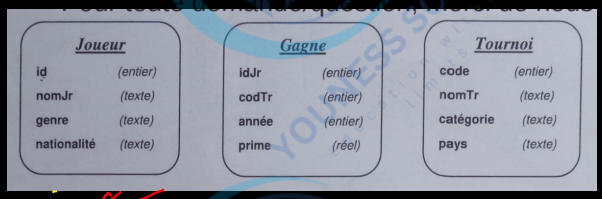



```
SELECT pays FROM Tournoi WHERE catégorie = 'ATP 500' OR catégorie = 'Master 1000' ;
```

(✓)

OR

```
SELECT pays, catégorie FROM Tournoi WHERE catégorie='ATP 500' or catégorie='Master 1000' ;
```




| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

→ Joueur

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

Tournoi

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Gagne

MAX

**Q.5** – Rédiger une requête SQL, qui permet d'obtenir les codes des tournois, la plus grande prime, la plus petite prime et la moyenne des primes de chaque tournoi, triés dans l'ordre décroissant de la moyenne des primes.

MIN

AVG

SELECT

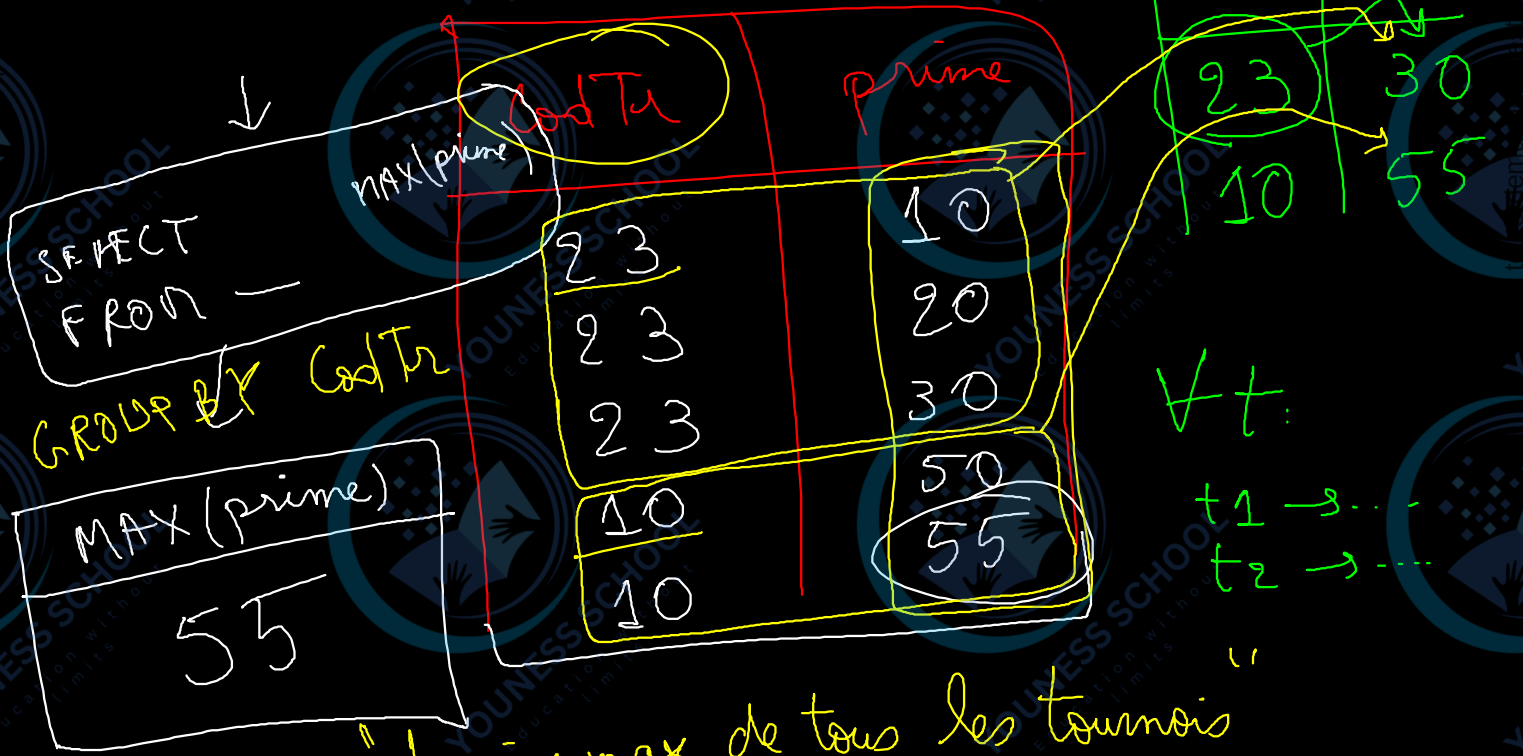
ORDER BY

GROUP BY

23 → ?
85 → ?
10 → ?
...

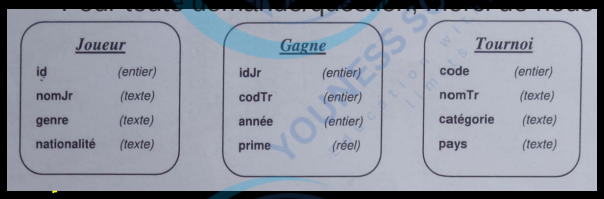
Gagne → CodTr
→ Prime

```
SELECT CodTr, MAX(prime), MIN(prime), AVG(prime) AS M
FROM Gagne
GROUP BY CodTr
ORDER BY M DESC ;
```




"La prime max de tous les tournois"

"La prime max pour chaque tournoi"




| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

→ Joueur

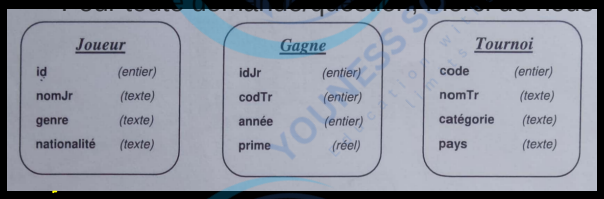
Tournoi

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Gagne

**Q.6** – Rédiger une requête SQL, qui permet d'obtenir les noms et genres des joueurs, le compte des tournois gagnés par chaque joueur et la somme totale des primes reçue par chaque joueur. La somme totale des primes reçue par chaque joueur doit être supérieure à **75** millions de dollars.




| id | nomJr | genre | nationalité |
|---|---|---|---|
| 2154 | Roger Federer | M | Suisse |
| 3432 | Serena Williams | F | États Unis |
| 2691 | Novak Djokovic | M | Serbie |
| 2374 | Rafael Nadal | M | Espagne |
| ... | ... | ... | ... |

→ Joueur

Tournoi

| code | nomTr | catégorie | pays |
|---|---|---|---|
| 23 | Roland-Garros | Grand Chelem | France |
| 174 | Doha | ATP 250 | Qatar |
| 10 | Open d'Australie | Grand Chelem | Australie |
| 59 | Sydney International | ATP 250 | Australie |
| 85 | Rotterdam | ATP 500 | Pays-Bas |
| 96 | Masters d'Indiana Wells | Masters 1000 | États Unis |
| ... | ... | ... | ... |

| idJr | codTr | année | prime |
|---|---|---|---|
| 2374 | 23 | 2019 | 2.30 |
| 2374 | 23 | 2020 | 1.60 |
| 2374 | 23 | 2021 | 1.40 |
| 3432 | 85 | 2018 | 2.50 |
| 2691 | 10 | 2020 | 4.12 |
| 2691 | 10 | 2021 | 2.75 |
| ... | ... | ... | ... |

Gagne

**Q.7** – Les tournois de la catégorie **Grand-Chelem** sont 4 : **'Open d'Australie'**, **'Roland-Garros'**, **'Wimbledon'** et **'US Open'**.

Rédiger une requête SQL, qui permet d'obtenir les années, les noms, les genres et les nationalités des joueurs qui ont gagné les 4 tournois du **Grand-Chelem** la même année, triés dans l'ordre décroissant des années.