$\mathcal{MPSI}$ 2

LYCEE CHARLEMAGNE
2020/2021

VENDREDI 4 DÉCEMBRE

IPT

$\mathcal{MPSI}$ 2 — Est ce cruel ? — IPT

Une base de données contient la table des horaires de trains qui rouleront demain en France.

Horaires					
Depart	HeureD	Arrivee	HeureA	Rame	Id
string(30)	time	string(30)	time	int(5)	int(5)

Écrivez les requêtes SQL pour les problèmes suivants :

A	Vérifiez s'il y a des trains qui arrivent avant d'être partis (on ne se méfie jamais des trains qui arrivent à l'heure et même avant).
B	Combien de trains partiront de Saint-Lazare après 14h00 pour arriver à Lisieux (trains directs).
C	Combien de trajets de Saint-Lazare à Pont-Leveque arrivant avant 16h00 (trains directs).
D	Quelle sera la réponse à `SELECT HeureA-HeureD AS D FROM Horaires WHERE Depart = "Montparnasse" AND Arrivee = "Brest" ORDER BY D DESC`
E	Combien de trajets de Saint-Lazare à Pont-Leveque (trains avec correspondance à Lisieux).
F	Quels seront les trajets réalisés par la rame 43235 demain après midi ?
G	Combien de rames seront en train de rouler demain à 15h30 ?
H	Combien de trains partiront demain d'une des six gares parisiennes (si vous n'en connaissez pas les noms, inventez).
I	Est il possible de faire un aller retour Paris Hendaye dans la journée ?
J	Existe-t-il un trajet dont la gare de départ est aussi la gare d'arrivée (comme sur le RER C parfois, ou la ligne de métro 7 bis).
K	Existe-t-il des trajets pour lesquels deux trains partent à la même heure (à un quart d'heure près) et arrivent à la même heure (à un quart d'heure près), entre les deux mêmes villes.
L	Les rames TGV sont repérées par un numéro de rame commençant par 5. Donnez la liste des trains qui relient Saint-Charles (gare de Marseille) à Perrache (gare de Lyon) qui ne sont pas de TGV.
M	Le premier train qui roule demain matin, il va de où à où ?
N	Demain, Jonas sera aux Sables d'Olonne (terminus), mais il voudra prendre le train. Où pourra-t-il aller (ligne directe, sans correspondance) en moins de trois heures de trajet ?
O	Gaspard est à Fontenay le Comte ; pas un train ne s'arrête, mais c'est normal, la ligne est déclassée. Mais en tendant bien l'oreille, il peut entendre les trains qui vont de Niort à Montparnasse (et vice versa). Demain, il en entendra passer combien ?
	un point par question, barème double pour les deux ou trois questions plus délicates

Un champ de type `time` est en fait numérique. On peut faire des tests du type `HeureD<HeureA`.
On peut calculer des différences : `HeureA-HeureD`, elles seront exprimées en minutes.

$\mathcal{MPSI}$ 2 — Scooby Doo by Dooooo — IPT

`Scooby`, `Velma` et `Fred` sont de retour.

```
def Scooby(n) :
....S = 0
....while n > 0 :
........S = 10*S+(n%10)
........n = n//10
....return(S)
```

```
def Velma(n) :
....L = [n]
....while n != Scooby(n) :
........n += Scooby(n)
........L.append(n)
....return(L)
```

Indiquez ce que fait `Scooby` pour les valeurs de `n` suivantes, en complétant l'exécution pas à pas du tableau
$n = 2021$, $n = 16261180$, $n = 9!$ 3 pt.

	S	n	test
étape			
⋮			

On invite ensuite Velma [1]. Indiquez le résultat de `Velma` pour n égal à 59, 79 et 296 (calculatrice autorisée) et 365. 4 pt.

On observe que le script appelle deux fois la fonction `Scooby`, ce qui est du gaspillage (même si `Python` vient de calculer `Scooby(18227)` à la ligne `while n != Scooby(n)`, il le recalcule à la ligne `n += Scooby(n)`. Modifiez le programme pour éviter ça. 1 pt.

On sait qu'il ne faudra pas lancer `Velma(196)`, ni `Velma(887)` et quelques autres (196, 887, 1675, 7436, 13783...[2]) ; pour eux, le programme ne s'arrête jamais. Faites que `Velma` s'arrête quand même si la longueur de `L` dépasse une valeur `N` imposée à l'avance. 2 pt.

Écrivez un script `Fred` qui prend en entrée deux nombres `a` et `b` et retourne la liste des entiers `n` entre `a` et `b` pour lesquels la liste `Velma(n)` est la plus longue. 3 pt.

$\mathcal{M}$ $\mathcal{P}$ $\mathcal{S}$ $\mathcal{I}$ 2  Ne videz pas de chopine sur le quai.  **IPT**

Vous gérez une gare de type terminus. Il y a des trains qui arrivent, et d'autres qui repartent.
Vous avez la liste des heures d'arrivées des trains, triée par ordre croissant.
Vous avez la liste des heures de départ des trains, triée aussi par ordre croissant.
Les heures sont en fait comptées en « nombre de minutes depuis 0 heure », le train de 7h20 est en fait le train de 440, d'accord ?
On estime que dès son arrivée en gare, un train peut être considéré comme vide et remis au départ quand vous voulez.
Mais le quai qu'il occupe entre son arrivée et son nouveau départ ne peut pas être utilisé par un autre train évidemment.
Il s'agit de vérifier si avec `n` quais vous pouvez satisfaire la demande.
Votre procédure prenant en entrée deux listes de même longueur `Arrivees` et `Departs` et l'entier `n` devra dire si vous avez effectivement assez de quais (elle retournera `True`). Si il y a un instant où la gare aura des trains sur tous ses quais et un nouveau train qui arrive, elle répondre False. De même si il est demandé de faire partir un nouveau train alors que vous n'avez aucune rame à quai, vous répondrez `False`.
Exemple :`Arrivees = [495 , 562 , 740 , 841 , 920 , 1264]`,
`Departs = [624 , 803 , 950 , 1047 , 1133 , 1320]`
On doit avoir au moins 3 quais [3] disponibles, avec par exemple cette utilisation :

	495	562	624	740	803	841	920	950	1047	1133	1264	1320
Quai A	a	=	=	=	d		a	=	=	=	d													
Quai B			a	=	=	=	=	=	d				a	=	=	=	=	=	d					
Quai C							a	=	=	=	=	=	=	=	d									

On supposera qu'il n'y a pas deux trains qui arrivent ou qui partent à la même heure. 8 pt.

C'est trop difficile pour vous ? Alors faites moi au moins les convertisseurs : 3 pt.

"986" donne "16 h 26"	"986" donne "4 h 26 pm"	"14 h 30" donne "870"
en fait `986` donne `16, 26`	en fait `986` donne `4, 26, pm`	en fait `14, 30` donne `870`
prend un entier `t` entre `0` et `1440` (minutes après minuit) et retourne deux entiers `h` et `m` (heure et minutes)	prend un entier `t` entre `0` et `1440` (minutes après minuit) et retourne deux entiers `h` et `m` et une chaine (heure, minutes, am/pm)	prend un couple h et m (heures et minutes) et retourne le temps écoulé depuis minuit

Et même si vous savez traiter la question du nombre de quais, traitez moi les convertisseurs.

$\mathcal{M}$ $\mathcal{P}$ $\mathcal{S}$ $\mathcal{I}$ 2  36 points  **IPT**

1. Véra en version française
2. nombres de Lychrel, suite A006960 de l'encyclopédie Sloane
3. un beau quai qui manque de chaises ?

$\mathcal{MPSI}$ 2	CORRECTION
LYCEE CHARLEMAGNE 2020/2021	IPT

A	Vérifiez s'il y a des trains qui arrivent avant d'être partis (on ne se méfie jamais des trains qui arrivent à l'heure et même avant).

	`SELECT *`
	`FROM Horaires`
	`WHERE HeureD>HeureA`

B	Combien de trains partiront de Saint-Lazare après 14h00 pour arriver à Lisieux (trains directs).

	`SELECT COUNT(*)`
	`FROM Horaires`
	`WHERE Depart = "Saint-Lazare"`
	`  AND Arrivee = "Lisieux"`
	`  AND HeureD > "14h00"`

C	Combien de trajets de Saint-Lazare à Pont-Leveque arrivant avant 16h00 (trains directs).

	`SELECT COUNT(*)`
	`FROM Horaires`
	`WHERE Depart="Saint-Lazare"`
	`  AND Arrivee = "Pont-Leveque"`
	`  AND HeureA < "16h00"`

D	Quelle sera la réponse à `SELECT HeureA-HeureD AS D`
	`FROM Horaires`
	`WHERE Depart = "Montparnasse" AND Arrivee = "Brest"`
	`ORDER BY D DESC`

	On étudie les trajets Paris-Brest. On calcule la durée de chaque trajet (qu'on note D). Et on affiche la liste de ces durées. De la plus longue à la plus courte (d'où le tri sur D déjà calculé).
E	Combien de trajets de Saint-Lazare à Pont-Leveque (trains avec correspondance à Lisieux).

	`SELECT COUNT(*)`
	`FROM (Horaires AS T1 JOIN Horaires AS T2`
	`  ON T1.Depart="Saint-Lazare"`
	`  AND T1.Arrivee = "Lisieux"`
	`  AND T2.Depart = "Lisieux"`
	`  AND T2.Arrivee = "Pont-Leveque"`
	`  AND T1.HeureA < T2.HeureD)`
	Il faut lire deux copies de la table horaire.
	Une pour le trajet Paris-Lisieux ; une pour le trajet Lisisux-Pont-Leveque.
	Et il faut quand que le deuxième train ne parte de Lisieux qu'après l'arrivée du premier.

F	Quels seront les trajets réalisés par la rame 43235 demain après midi ?

	`SELECT Depart, Arrivee`
	`FROM Horaires`
	`WHERE Rame = 43235`
	`AND HeureD >= 12h00`
	
	Il y a la condition « après midi ».
	Et le numéro de rame est un numéro, c'est donc un test `=34235` et pas `="43235"`.

G	Combien de rames seront en train de rouler demain à 15h30 ?

	`SELECT COUNT(*)`
	`FROM Horaires`
	`WHERE (HeureD < "15h30"`
	`  AND HeureA > "15h30")`

H — Combien de trains partiront demain d'une des six gares parisiennes (si vous n'en connaissez pas les noms, inventez).

```
SELECT COUNT(*)
FROM Horaires
WHERE Depart IN ("Saint-Lazare", "Montparnasse", "Austerlitz", "Gare de Lyon",
"Garde de l'Est", "Gare du Nord")
```

I — Est il possible de faire un aller retour Paris Hendaye dans la journée ?

```
SELECT COUNT(*)
FROM Horaires AS Aller JOIN Horarires AS Retour
WHERE Aller.Depart = "Montparnasse"
      AND Aller.Arrivee = "Hendaye"
      AND Retour.Depart = "Hendaye"
      AND Retour.Arrivee = "Montparnasse"
      AND Retour.Depart > Aller.Arrivee
```

Oui, le même schéma que tout à l'heure... Mais là, c'est cinq heures de train aller et autant retour. On espère qu'il ne faut pas en plus envisager des correspondances...

J — Existe-t-il un trajet dont la gare de départ est aussi la gare d'arrivée (comme sur le RER C parfois, ou la ligne de métro 7 bis).

```
SELECT *
FROM Horaires
WHERE Depart = Arrivee
```

K — Existe-t-il des trajets pour lesquels deux trains partent à la même heure (à un quart d'heure près) et arrivent à la même heure (à un quart d'heure près), entre les deux mêmes villes.

```
SELECT *
FROM Horaires AS Premier JOIN Horarires AS Second
WHERE Premier.Depart=Second.Depart
      AND Premier.Arrivee=Second.Arrivee
      AND ABS(Premier.Depart-Second.Depart)<15
      AND ABS(Premier.Arrivee-Second.Arrivee)<15
      AND NOT(Premier.Id=Second.Id)
```

Deux conditions pour « le même trajet ».
Deux conditions sur la différence des horaires.
Mais ne pas oublier non plus qu'il ne faut pas considérer le même train !

L — Les rames TGV sont repérées par un numéro de rame commençant par 5. Donnez la liste des trains qui relient Saint-Charles (gare de Marseille) à Perrache (gare de Lyon) qui ne sont pas de TGV.

```
SELECT *
FROM Horaires
WHERE Depart = "Saint-Charles"
      AND Arrivee = "Perrache"
      AND NOT(Rame LIKE "5%")
```

M — Le premier train qui roule demain matin, il va de où à où ?

```
SELECT Depart, Arrivee
FROM Horaires
ORDER BY HeureD LIMIT 1
```

Faire un tri pour ne garder que le minimum, il faut oser !

N — Demain, Jonas sera aux Sables d'Olonne (terminus), mais il voudra prendre le train. Où pourra-t-il aller (ligne directe, sans correspondance) en moins de trois heures de trajet ?

```
SELECT Arrivee
FROM Horaires
WHERE Depart = "Les Sables d'Olonne"
      AND HeureA-HeureD<180
```

O — Gaspard est à Fontenay le Comte ; pas un train ne s'arrête, mais c'est normal, la ligne est déclassée. Mais en tendant bien l'oreille, il peut entendre les trains qui vont de Niort à Montparnasse (et vice versa). Demain, il en entendra passer combien ?

```
SELECT COUNT(*)
FROM Horaires
WHERE (Depart = "Niort" AND Arrivee = "Montparnasse")
      OR (Depart = "Montparnasse" AND Arrivee = "Niort")
```

	S	n	test	n%10	10*S+(n%10)	n//10
étape	0	2021	True	1	1	202
	1	202	True	2	12	20
	12	20	True	0	120	2
	120	2	True	2	1202	0
	1202	0	False			

On sort en retournant la valeur 1202.

	S	n	test	n%10	10*S+(n%10)	n//10
étape	0	1621180	True	0	0	162118
	0	162118	True	8	8	16211
	8	16211	True	1	81	1621
	81	1621	True	1	811	162
	811	162	True	2	8112	16
	8112	16	True	6	81126	1
	81126	1	True	1	811261	0
	811261	0	False			

On retourne... justement l'entier retourné.

	S	n	test	n%10	10*S+(n%10)	n//10
étape	0	362880	True	0	0	36288
		36288	True	8	8	3628
		3628	True	8	88	362
		362	True	2	882	36
		36	True	6	8826	3
		3	True	3	88263	0
		0	False			

Que fait ensuite Vera ? Elle travaille sur un entier n et le modifie.
Tant que l'entier n'est pas égal à son renversé (tant que ce n'est pas un palindrome),
Elle fait la somme de l'entier et de son renversé.
Et elle le met dans la liste L.

La liste L va donc contenir les éléments successifs d'une suite « u_{n+1}= u_n+son renversé », jusqu'à tomber sur un palindrome.

n	Scooby(n)	n !=Scooby(n)	n+Scooby(n)	L
59	95	True	154	[59, 154]
154	451	True	605	[59, 154, 605]
605	506	True	1111	[59, 154, 605, 1111]
1111	1111	False		
[59, 154, 605, 1111]				

n	Scooby(n)	n !=Scooby(n)	n+Scooby(n)	L
79	97	True	176	[79, 176]
176	671	True	847	[79, 176, 847]
847	748	True	1595	[79, 176, 847, 1595]
1595	5951	True	7546	[79, 176, 847, 1595, 7546]
7546	6457	True	14003	[79, 176, 847, 1595, 7546, 14003]
14003	30041	True	44044	[79, 176, 847, 1595, 7546, 14003, 44044]
44044	44044	False		
[79, 176, 847, 1595, 7546, 14003, 44044]				

Avec 296, la liste produite est `[296, 988, 1877, 9658, 18227, 90508, 171017, 881188]`.

Et on peut craquer avec 365 :

```
[365, 928, 1757, 9328, 17567, 94138, 177287, 960058, 1810127, 9020308, 17050517, 88555588]
```

Pour ne pas calculer deux fois `Scooby(n)`

```
def Velma(n) :
....L = [n]
....Sc = Scooby(n)
....while n != Sc :
........n += Sc
........L.append(n)
........Sc = Scooby(n)
....return(L)
```

Pour arrêter ? On se demande quelle est la condition d'arrêt : • soit `n` est égal à son `Scooby`
• soit `len(L)` a atteint `N`
C'est un « ou ». La condition pour poursuivre est donc sa négation, c'est un « et ».

```
def Velma(n) :
....L = [n]
....Sc = Scooby(n)
....while (n != Sc) and (len(L)<N) :
........n += Sc
........L.append(n)
........Sc = Scooby(n)
....return(L)
```

Avec `N` égal à 25, voici `Velma(196)` :
[196, 887, 1675, 7436, 13783, 52514, 94039, 187088, 1067869, 10755470, 18211171, 35322452, 60744805, 111589511, 227574622, 454050344, 897100798, 1794102596, 8746117567, 16403234045, 70446464506, 130992928913, 450822227944, 900544455998, 1800098901007]

Remarque : On ne va pas s'arrêter ? Peut on même prouver qu'on ne s'arrêtera pas ? Attention, une vraie preuve, et pas « rien trouvé même avec N=100 » [a].

a. oui : 1315889733122632059256287274205914623024788951 3, pas encore un palindrome, c'est désespérant !

Et `Fred` ? On parcourt en impératif `range(a, b)`.
On note `Record` le record à battre. Et `L` la liste des entiers réalisant ce record (il peu y en avoir plusieurs).
Pour chaque n, • si il dépasse le record en cours : on efface la liste `L`,
on y met n tout seul,
et on définit le nouveau record à battre
• si n atteint le record en cours : on le colle dans la liste des recormen (ou recordnumb)
• sinon, on ne fait rien, tant pis

```
def Fred(a, b) :
....Record = 0
....L = [ ]
....for n in range(a, b) :
........Score = len(Velma(n))
........if Score > Record :
............L = [n]
............Record = Score
........elif Score == Record :
............L.append(n)
....return(Record, L)
```

```
def Fred(a, b) :
....Record = 0
....L = [ ]
....for n in range(a, b) :
........Score = len(Velma(n))
........if Score > Record :
............L = [ ]
............Record = Score
........if Score == Record :
............L.append(n)
....return(Record, L)
```

Voyez vous la différence entre ces deux versions ?

$\mathcal{M}$	$\mathcal{P}$	$\mathcal{S}$	$\mathcal{I}$	2

Les trains à quai.

IPT

On manipule une variable des quais nécessaires : `NQuais`.
Au départ, elle est à `0`.
On parcourt la liste des trains à l'arrivée.

Quand un train arrive, on augmente `NQuais` de 1.
Mais si entre-temps un train part, on enlève `1` à `NQuais`.
Comment savoir si la prochaine action est un train qui arrive ou un train qui part ?

C'est pourquoi la boucle ne sera pas impérative, mais en `while`.
Tant qu'on n'a pas vu arriver tous les trains, on regarde ce qu'il se passe.
On va incrémenter deux compteurs : celui de lecture de la liste des trains qui arrivent et celui de la liste des trains qui partent.

Déjà sans les tests, en comptant les quais :

```
def Quais(Arrivees, Departs,n) :
....NTrains = len(Arrivees)#combien de trains aujourd'hui ?
....ca, cd = 0, 0 #initialisation des compteurs
....while ca < n :
........if Arrivees[ca] < Departs[cb] : #procahin événement : arrivée ou départ
............NQuais += 1 #une arrivée, il faut un quai de plus
............ca += 1 #et on regarde la prochaine arrivée
........else : #un train va partir
............NQuais -=1 #il libére un quai
............cd +=1 #on regarde le train suivant
```

Et avec les tests • a-t-on assez de quais
• avait on un train à quai à faire partir

```
def Quais(Arrivees, Departs,n) :
....NTrains = len(Arrivees)
....ca, cd = 0, 0
....while ca < n :
........if Arrivees[ca] < Departs[cb] :
............NQuais += 1
............ca += 1
............if NQuais > n :
................return(False)
........else :
............NQuais -=1 #on libère un quai
............cd +=1
............if NQuais < 0 : #mais il n'y avait pas de train disponible !
................return(False)
............if cd > n :
................return(....) #oui, que se passe-t-il là ?
....return(True) #dernier départ, tout s'est bien passé
```

Source :
site Techie Delight
`https://www.techiedelight.com/minimum-number-of-platforms-needed-avoid-delay-arrival-train/`
recherche Google : « Techie Delight train »
adapté par Eric Detrez (prof d'info et IPT à Lille) `https://www.faidherbe.org/~informatique/`

Version Techie Delight

```
# Function to find minimum number of platforms needed in the
# station so to avoid any delay in arrival of any train.
def minPlatforms(arrival, departure) :
....# maintains the count of trains
....count = 0
....# stores minimum platforms needed
....platforms = 0
....# take two indices for arrival and departure time
```

```
....i = j = 0
....# run till train is scheduled to arrive
....while i < len(arrival) :
........# if train is scheduled to arrive next
........if arrival[i] < departure[j] :
............# increase the count of trains and update minimum
............# platforms if required
............count = count + 1
............platforms = max(platforms, count)
............# move the pointer to next arrival
............i = i + 1
........# if train is scheduled to depart next i.e.
........# (departure[j] < arrival[i]), decrease the count of trains
........# and move pointer j to next departure
........# If two trains are arriving and departing at the same time, i.e.
........# (arrival[i] == departure[j]) depart the train first
........else :
............count = count - 1
............j = j + 1
....return platforms

# Find minimum number of platforms needed in the station to avoid any
# delay in arrival of any train
if __name__ == '__main__' :
....arrival = [2.00, 2.10, 3.00, 3.20, 3.50, 5.00]
....departure = [2.30, 3.40, 3.20, 4.30, 4.00, 5.20]
....# sort arrival time of trains
....arrival.sort()
....# sort departure time of trains
....departure.sort()
....print("Minimum platforms needed is", minPlatforms(arrival, departure))
```

Version Detrez donnant le nombre de quais

```
def nb_quais ( arrivees , departs ) :
....n = len ( arrivees )
....ia = 0
....id = 0
....nb = 0
....nb_max = 0
....while ia < n :
........if arrivees [ia] < departs [ib ] :
............nb = nb + 1
............ia = ia + 1
............if nb > nb_max :
................nb_max = nb
........else :
............nb = nb - 1
............ib = ib + 1
....return(nb_max)
```