

Devoir Surveillé n°1 de NSI

Exercice 1 – Histoire de l’informatique (2 points) :

Voici une liste d’événements historiques rencontrés dans notre Timeline :

- A. Création de MS-DOS
- B. Début RGPD
- C. Définition des protocoles TCP/IP
- D. Noyau Linux
- E. Ouverture des MP2I
- F. Premier circuit intégré

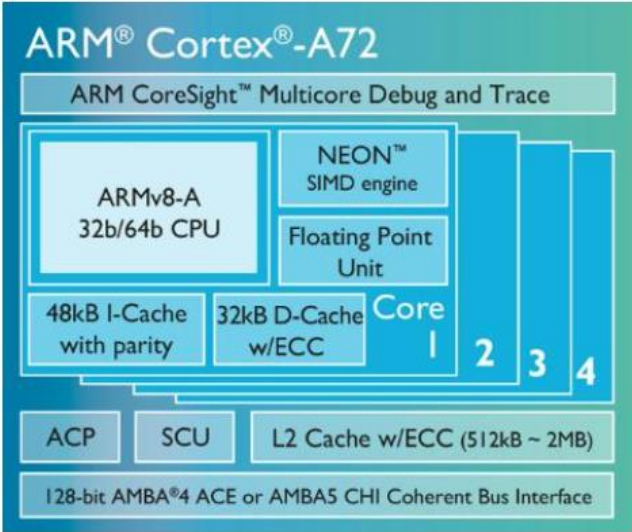
1. Classer ces 6 événements par ordre chronologique :
2. Placer ces événements dans la bonne décennie (une réponse par case) :

1950	1960	1970	1980	1990	2000	2010	2020

3. Donner un événement ayant eu lieu avant 1950 en précisant la décennie :

Exercice 2 – Raspberry (6 points) :

Voici un extrait des caractéristiques du dernier Raspberry PI 4 :



ARM® Cortex®-A72

ARM CoreSight™ Multicore Debug and Trace

ARMv8-A 32b/64b CPU

NEON™ SIMD engine

Floating Point Unit

48kB I-Cache with parity

32kB D-Cache w/ECC

Core 1 2 3 4

ACP SCU L2 Cache w/ECC (512kB ~ 2MB)

128-bit AMBA®4 ACE or AMBA5 CHI Coherent Bus Interface

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H.264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

1. Donner des informations/explications sur les caractéristiques suivantes (2 points) :

- a) SDRAM :
- b) 802.11ac wireless :
- c) Gigabit Ethernet :
- d) Floating Point Unit :

Nous nous intéressons désormais au SoC BCM2711.**2. Que signifie SoC (1 point) :**

- a) En anglais :
- b) En français :

3. Donner quatre caractéristiques d'un SoC (1 point) :

- a)
- b)
- c)
- d)

Pour calculer le nombre d'opérations par seconde (FLOPS) du SoC, on utilise la formule suivante :

$\text{FLOPS} = \text{nb}_{\text{cœur}} * \text{freq} * \text{nb}_{\text{flops/cycle}}$

4. Sachant que le SoC réalise 4 flops par cycle d'horloge (2 points) :

- a) De combien de cœur(s) est constitué le SoC de ce Raspberry ?
- b) Calculer le nombre d'opérations du Raspberry :
.....
- c) *Comparez* le résultat aux 6 Tflops de la Xbox one X :
.....

Exercice 3 – Service de streaming musical (6 points) :

!!! Attention, cet exercice est différent de l'interrogation du chapitre 5 !!!

Afin de lancer un nouveau service de streaming de musique, vous devez construire une base de données pour les morceaux de votre catalogue. Pour l'instant vous disposez d'une seule table avec les informations des morceaux.

Voici un extrait de cette table :

Titre	Durée	Artiste	Album	Piste	CD	Année
Astronomy	384	Blue Öyster Cult	Secret Treaties	8	1	1974
Stone Cold Crazy	136	Queen	Sheer Heart Attack	8	1	1974
Under Pressure	242	Queen and David Bowie	Hot Space	11	2	1982
The Outlaw Torn	589	Metallica	Load	14	1	1996
Fuel	270	Metallica	Reload	1	1	1997
The Memory Remains	279	Metallica and Marianne Faithfull	Reload	2	1	1997
Astronomy	398	Metallica	Garage Inc.	8	1	1998
Stone Cold Crazy	139	Metallica	Garage Inc.	11	2	1998
Fuel	276	Metallica and the San Francisco Symphony	S&M	6	1	1999
The Outlaw Torn	599	Metallica and the San Francisco Symphony	S&M	6	2	1999

Cette table ne convient pas vraiment pour faire une base de données.

1. Expliquer pourquoi aucune des colonnes ne peut pas servir de clef primaire en justifiant (0,5 point) :

.....

.....

2. Donner deux raisons expliquant pourquoi cette table est problématique si on veut rajouter des informations sur les artistes, comme leur nationalité ? (1 point)

.....

.....

.....

3. Quel est le problème si on souhaite chercher les morceaux d'un artiste ? (0,5 point)

.....

.....

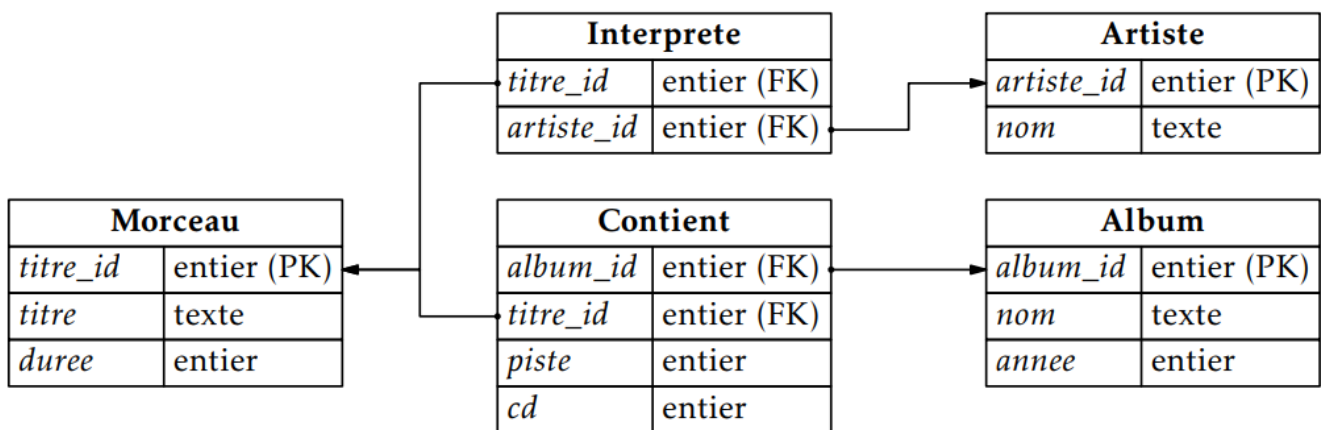
Un ami vous suggère d'utiliser le schéma suivant :

- ✓ **Morceau**(titre_id, titre, durée, #artiste_id, album, piste, cd, année)
- ✓ **Artiste**(artiste_id, nom)

4. Expliquer pourquoi cette représentation ne permet toujours pas de gérer les morceaux fait par deux artistes différents (0,5 point) :

.....

.....

Vous arrivez finalement au schéma suivant :

5. a) Comment appelle-t-on les schémas indiquant les PK & FK (0,5 point) ?
- b) Que signifie « FK » (français et anglais) ? (0,5 point)
- c) Revenir au schéma conceptuel en indiquant les cardinalités (1 point) :

6. Compléter les tables à l'aide des informations déjà disponibles. Un des morceaux n'a pas été remis. Si les noms dépassent, mettre uniquement le début (1 point) :

titre_id	titre	duree	titre_id	artiste_id	artiste_id	nom
519	Astronomy		519	25		
1219	Astronomy		1219	154		
316	Stone Cold Crazy	136	1319			
1319	Stone Cold Crazy	139	1298	154		
1298		270	1570	154		
1570			1570	318		
401			1125	154		
1125	The Outlaw Torn	589	1591	154		
1591	The Outlaw Torn	599	1591	318		
			316	79		
			401	79		
			401	108		

7. Expliquer pourquoi le couple (titre_id, artiste_id) peut servir de clef primaire à Interprete (0,5 point) :

.....

.....

.....

.....

Exercice 4 – Epreuve Pratique 1.2 (2 points) :

On souhaite programmer une fonction donnant la distance la plus courte entre un point de départ et une liste de points. Les points sont tous à coordonnées entières.

Les points sont donnés sous la forme d'un tuple de deux entiers.

La liste des points à traiter est donc un tableau de tuples.

On rappelle que la distance entre deux points du plan de coordonnées (x ; y) et (x' ; y') est donnée par la formule :

$$d = \sqrt{(x - x')^2 + (y - y')^2}$$

On importe pour cela la fonction racine carrée (sqrt) du module math de Python.

On dispose d'une fonction **distance** et d'une fonction **plus_courte_distance** :

```
from math import sqrt # import de la fonction racine carrée

def distance(point1, point2):
    """ Calcule et renvoie la distance entre deux points. """
    return sqrt((.....)**2 + (.....)**2)

assert distance((1, 0), (5, 3)) == 5.0, "erreur de calcul"

def plus_courte_distance(tab, depart):
    assert .....
    assert .....

    """ Renvoie le point du tableau tab se trouvant à la plus courte distance du point
    depart. """

    point = tab[0]
    min_dist = .....

    for i in range (1, .....):
        if distance(tab[i], depart).....:
            point = .....
            min_dist = .....

    return min_dist

assert plus_courte_distance([(7, 9), (2, 5), (5, 3)], (0, 0)) == (2, 5), "erreur"
```

Compléter le code de ces deux fonctions et ajouter deux déclarations (assert) à la fonction distance permettant de vérifier la ou les préconditions.

Exercice 5 – Epreuve Pratique 29.2 (2 points) :

On affecte à chaque lettre de l'alphabet un code selon les tableaux ci-dessous :

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Pour un mot donné, on détermine d'une part son *code alphabétique concaténé*, obtenu par la juxtaposition des codes de chacun de ses caractères, et d'autre part, son *code additionné*, qui est la somme des codes de chacun de ses caractères.

Par ailleurs, on dit que ce mot est « *parfait* » si le code additionné divise le code concaténé.

Exemples :

- ✓ Pour le mot "PAUL", le code concaténé est la chaîne 1612112, soit l'entier 1 612 112. Son code additionné est l'entier 50 car $16 + 1 + 21 + 12 = 50$. 50 ne divise pas l'entier 1 612 112 ; par conséquent, le mot "PAUL" n'est pas parfait.
- ✓ Pour le mot "ALAIN", le code concaténé est la chaîne 1121914, soit l'entier 1 121 914. Le code additionné est l'entier 37 car $1 + 12 + 1 + 9 + 14 = 37$. 37 divise l'entier 1 121 914 ; par conséquent, le mot "ALAIN" est parfait.

Compléter la fonction `est_parfait` page suivante qui prend comme argument une chaîne de caractères `mot` (en lettres majuscules) et qui renvoie le code alphabétique concaténé, le code additionné de mot, ainsi qu'un booléen qui indique si mot est parfait ou pas.

```
dico = {"A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9, "J":10, "K":11, "L":12, "M":13, "N":14, "O":15, "P":16, "Q":17, "R":18, "S":19, "T":20, "U":21, "V":22, "W":23, "X":24, "Y":25, "Z":26}
```

```
def est_parfait(mot) :
```

```
    #mot est une chaîne de caractères (en lettres majuscules)
```

```
    code_c = ""
```

```
    code_a = .....
```

```
    for c in mot :
```

```
        code_c = code_c + .....
```

```
        code_a = .....
```

```
    code_c = int(code_c)
```

```
    if ..... :
```

```
        mot_est_parfait = True
```

```
    else :
```

```
        mot_est_parfait = False
```

```
    return [code_a, code_c, mot_est_parfait]
```

Exemples :

```
>>> est_parfait("PAUL")  
[50, 1612112, False]
```

```
>>> est_parfait("ALAIN")  
[37, 1121914, True]
```