Nom : . . . . . .

Exercice 1 *4 points*

On a stocké dans une variable `musiciens` des renseignements concernant des musiciens de jazz : leur identifiant (`idmus`), leur nom (`nom`), leur prénom (`prenom`), leur instrument (`instru`), l'identifiant de leur groupe (`idgrp`) et leur nombre de concerts (`nb_concerts`).
La variable `musiciens` est un tableau (= une liste) de dictionnaires.
Aperçu du contenu de la variable :

```
>>> print(musiciens)
 [{'idmus': 12, 'nom': 'Parker', 'prenom': 'Charlie',
 'instru': 'trompette', 'idgrp' : 96, 'nb_concerts': 5},
 {'idmus': 13, 'nom': 'Parker', 'prenom': 'Charlie',
 'instru': 'trombone', 'idgrp' : 25, 'nb_concerts': 9},
 {'idmus': 58, 'nom': 'Dufler', 'prenom': 'Candy',
 'instru': 'saxophone', 'idgrp' : 96, 'nb_concerts': 4},
 {'idmus': 97, 'nom': 'Miles', 'prenom': 'Davis',
 'instru': 'saxophone', 'idgrp' : 87, 'nb_concerts': 2},
 ...
 ]
```

Écrire la fonction `recherche_nom` ayant pour unique paramètre un tableau de dictionnaires (comme `musiciens` présenté précédemment) renvoyant un tableau contenant le nom de tous les musiciens ayant participé à au moins 4 concerts.

Exercice 2 *9 points*

1. Avec la commande `ps -aef` on obtient l'affichage suivant :

PID	PPID	C	STIME	TTY	TIME	CMD
8600	2	0	17 :38	?	00 :00 :00	[kworker/u2 :0-fl]
8859	2	0	17 :40	?	00 :00 :00	[kworker/0 :1-eve]
8866	2	0	17 :40	?	00 :00 :00	[kworker/0 :10-ev]
8867	2	0	17 :40	?	00 :00 :00	[kworker/0 :11-ev]
8887	6217	0	17 :40	pts/0	00 :00 :00	bash
9562	2	0	17 :45	?	00 :00 :00	[kworker/u2 :1-ev]
9594	2	0	17 :45	?	00 :00 :00	[kworker/0 :0-eve]
9617	8887	21	17 :46	pts/0	00 :00 :06	/usr/lib/firefox/firefox
9657	9617	17	17 :46	pts/0	00 :00 :04	/usr/lib/firefox/firefox - contentproc -childID
9697	9617	4	17 :46	pts/0	00 :00 :01	/usr/lib/firefox/firefox - contentproc -childID
9750	9617	3	17 :46	pts/0	00 :00 :00	/usr/lib/firefox/firefox - contentproc -childID
9794	9617	11	17 :46	pts/0	00 :00 :00	/usr/lib/firefox/firefox - contentproc -childID
9795	9794	0	17 :46	pts/0	00 :00 :00	/usr/lib/firefox/firefox
9802	7441	0	17 :46	pts/2	00 :00 :00	ps -aef

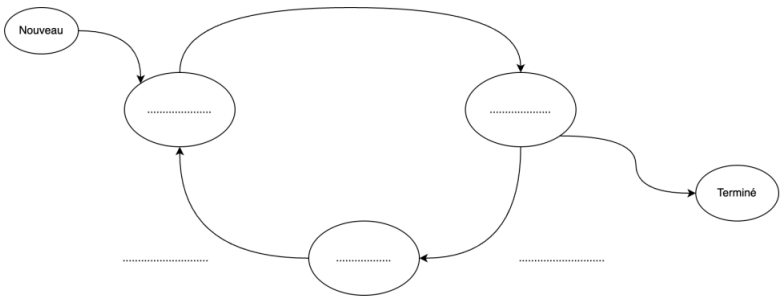
On rappelle que :

- PID = Identifiant d'un processus (Process Identification)
- PPID = Identifiant du processus parent d'un processus (Parent Process Identification)

a. Donner sous forme d'un arbre de PID la hiérarchie des processus liés à `firefox`.

b. La commande `kill` permet de supprimer un processus à l'aide de son PID (par exemple `kill 8600`). Indiquer la commande qui permettra de supprimer tous les processus liés à `firefox` et uniquement cela.

2. **a.** Compléter le schéma ci-dessous avec les termes suivants concernant l'ordonnancement des processus. *Élu*, *En attente*, *Prêt*, *Blocage*, *Déblocage*.



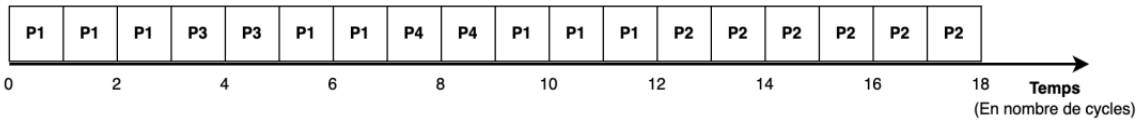
On donne dans le tableau ci-dessous quatre processus qui doivent être exécutés par un processeur. Chaque processus a un instant d'arrivée et une durée, donnés en nombre de cycles du processeur.

Processus	P1	P2	P3	P4
Instant d'arrivée	0	2	3	7
Durée	8	6	2	2

Les processus sont placés dans une file d'attente en fonction de leur instant d'arrivée. On se propose d'ordonnancer ces quatre processus avec la méthode suivante :

- Parmi les processus présents en liste d'attente, l'ordonnanceur choisit celui dont la durée restante est la plus courte ;
- Le processeur exécute un cycle de ce processus puis l'ordonnanceur désigne de nouveau le processus dont la durée restante est la plus courte ;
- En cas d'égalité de temps restant entre plusieurs processus, celui choisi sera celui dont l'instant d'arrivée est le plus ancien ;
- Tout ceci jusqu'à épuisement des processus en liste d'attente.

On donne en exemple ci-dessous, l'ordonnancement des quatre processus de l'exemple précédent suivant l'algorithme ci-dessus.

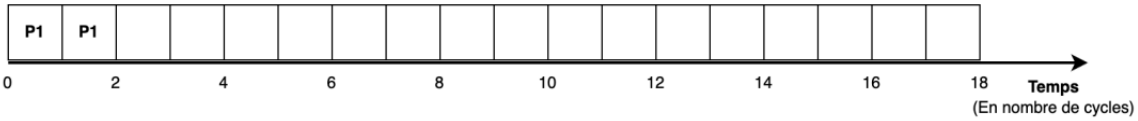


On définit le temps d'exécution d'un processus comme la différence entre son instant de terminaison et son instant d'arrivée.

b. Calculer la moyenne des temps d'exécution des quatre processus.

On se propose de modifier l'ordonnancement des processus. L'algorithme reste identique à celui présenté précédemment mais au lieu d'exécuter un seul cycle, le processeur exécutera à chaque fois deux cycles du processus choisi. En cas d'égalité de temps restant, l'ordonnanceur départagera toujours en fonction de l'instant d'arrivée.

c. Compléter le schéma ci-dessous donnant le nouvel ordonnancement des quatre processus.



d. Calculer la nouvelle moyenne des temps d'exécution des quatre processus et indiquer si cet ordonnancement est plus performant que le précédent.

Exercice 3 *3 points*

On considère trois processus P_1, P_2, P_3 et trois ressources R, S, T. Voilà ce qu'on sait sur l'utilisation des ressources par les processus :

- P_1 obtient la ressource R puis demande la ressource S.
- P_2 obtient la ressource T puis demande la ressource R.
- P_3 obtient la ressource S puis demande la ressource T.

Montrer que cette situation risque d'aboutir à un interblocage.

Exercice 4 *4 points*

On considère 4 processus C_1, C_2, C_3 et C_4 placés dans cet ordre dans la **file** d'attente de l'ordonnanceur.

- Les temps d'exécution totaux de C_1, C_2, C_3 et C_4 sont respectivement 60 ms, 150 ms, 110 ms et 60 ms.
- Après 40 ms d'exécution, le processus C_1 demande une opération d'écriture disque, opération qui dure 290 ms. Pendant cette opération d'écriture, le processus C_1 passe à l'état bloqué.
- Après 70 ms d'exécution, le processus C_3 demande une opération d'écriture disque, opération qui dure 40 ms. Pendant cette opération d'écriture, le processus C_3 passe à l'état bloqué.

Sur la frise chronologique ci-dessous, les états du processus C_2 sont donnés.
Compléter la frise avec les états des processus C_1, C_3 et C_4.