BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2021

NUMÉRIQUE et SCIENCES INFORMATIQUES

Jour 1

Durée de l'épreuve : 3 heures 30

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 12 pages numérotées de 1/12 à 12/12.

Le candidat traite au choix 3 exercices parmi les 5 exercices proposés

Chaque exercice est noté sur 4 points.

Exercice 3 (4 points).

Cet exercice porte sur les tableaux et sur la programmation de base en Python.

On rappelle que `len` est une fonction qui prend un tableau en paramètre et renvoie sa longueur. C'est-à-dire le nombre d'éléments présents dans le tableau.
Exemple : `len([12, 54, 34, 57])` vaut 4.

Le but de cet exercice est de programmer différentes réductions pour un site de vente de vêtements en ligne.
On rappelle que si le prix d'un article avant réduction est de x euros,

- son prix vaut $0,5x$ si on lui applique une réduction de 50%,
- son prix vaut $0,6x$ si on lui applique une réduction de 40%,
- son prix vaut $0,7x$ si on lui applique une réduction de 30%,
- son prix vaut $0,8x$ si on lui applique une réduction de 20%,
- son prix vaut $0,9x$ si on lui applique une réduction de 10%.

Dans le système informatique du site de vente, l'ensemble des articles qu'un client veut acheter, appelé *panier*, est modélisé par un tableau de flottants.
Par exemple, si un client veut acheter un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, le système informatique aura le tableau suivant :

`tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]`.

1. (a) Écrire une fonction Python `total_hors_reduction` ayant pour argument le tableau des prix des articles du panier d'un client et renvoyant le total des prix de ces articles.

   (b) Le site de vente propose la promotion suivante comme offre de bienvenue : 20% de réduction sur le premier article de la liste, 30% de réduction sur le deuxième article de la liste (s'il y a au moins deux articles) et aucune réduction sur le reste des articles (s'il y en a).
   Recopier sur la copie et compléter la fonction Python `offre_bienvenue` prenant en paramètre le tableau `tab` des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre de bienvenue.

```
def offre_bienvenue(tab):
    """ tableau -> float """
    somme=0
    longueur=len(tab)
    if longueur > 0 :
        somme=tab[0]*...
    if longueur > 1 :
        somme=somme + ...
    if longueur > 2 :
        for i in range(2,longueur):
            somme=...
    return ...
```

Pour toute la suite de l'exercice, on pourra utiliser la fonction `total_hors_reduction` même si la question 1 n'a pas été traitée.

2. Lors de la période des soldes, le site de vente propose les réductions suivantes :
   - si le panier contient 5 articles ou plus, une réduction globale de 50%,
   - si le panier contient 4 articles, une réduction globale de 40%,
   - si le panier contient 3 articles, une réduction globale de 30%,
   - si le panier contient 2 articles, une réduction globale de 20%,
   - si le panier contient 1 article, une réduction globale de 10%.

   Proposer une fonction Python `prix_solde` ayant pour argument le tableau `tab` des prix des articles du panier d'un client et renvoyant le total des prix de ces articles lorsqu'on leur applique la réduction des soldes.

3. (a) Écrire une fonction `minimum` qui prend en paramètre un tableau `tab` de nombres et renvoie la valeur minimum présente dans le tableau.

   (b) Pour ses bons clients, le site de vente propose une offre promotionnelle, à partir de 2 articles achetés, l'article le moins cher des articles commandés est offert.

   Écrire une fonction Python `offre_bon_client` ayant pour paramètre le tableau des prix des articles du panier d'un client et renvoyant le total à payer lorsqu'on leur applique l'offre bon client.

4. Afin de diminuer le stock de ses articles dans ses entrepôts, l'entreprise imagine faire l'offre suivante à ses clients : en suivant l'ordre des articles dans le panier du client, elle considère les 3 premiers articles et offre le moins cher, puis les 3 suivants et offre le moins cher et ainsi de suite jusqu'à ce qu'il reste au plus 2 articles qui n'ont alors droit à aucune réduction.

   Exemple : Si le panier du client contient un pantalon à 30,50 euros, un tee-shirt à 15 euros, une paire de chaussettes à 6 euros, une jupe à 20 euros, une paire de collants à 5 euros, une robe à 35 euros et un short à 10,50 euros, ce panier est représenté par le tableau suivant :

   ```
   tab = [30.5, 15.0, 6.0, 20.0, 5.0, 35.0, 10.5]
   ```

   Pour le premier groupe (le pantalon à 30,50 euros, le tee-shirt à 15 euros, la paire de chaussettes à 6 euros), l'article le moins cher, la paire de chaussettes à 6 euros, est offert. Pour le second groupe (la jupe à 20 euros, la paire de collants à 5 euros, la robe à 35 euros), la paire de collants à 5 euros est offerte.

   Donc le total après promotion de déstockage est 111 euros.

   On constate que le prix après promotion de déstockage dépend de l'ordre dans lequel se présentent les articles dans le panier.

   (a) Proposer un panier contenant les mêmes articles que ceux de l'exemple mais ayant un prix après promotion de déstockage différent de 111 euros.

   (b) Proposer un panier contenant les mêmes articles mais ayant le prix après promotion de déstockage le plus bas possible.

   (c) Une fois ses articles choisis, quel algorithme le client peut-il utiliser pour modifier son panier afin de s'assurer qu'il obtiendra le prix après promotion de déstockage le plus bas possible ? On ne demande pas d'écrire cet algorithme.

EXERCICE 4 (4 points)

Cet exercice porte sur les systèmes d'exploitation : gestion des processus et des ressources.

Les parties A et B peuvent être traitées indépendamment.

Partie A :

Dans un bureau d'architectes, on dispose de certaines ressources qui ne peuvent être utilisées simultanément par plus d'un processus, comme l'imprimante, la table traçante, le modem. Chaque programme, lorsqu'il s'exécute, demande l'allocation des ressources qui lui sont nécessaires. Lorsqu'il a fini de s'exécuter, il libère ses ressources.

Programme 1	Programme 2	Programme 3
demander (table traçante)	demander (modem)	demander (imprimante)
demander (modem)	demander (imprimante)	demander (table traçante)
exécution	exécution	exécution
libérer (modem)	libérer (imprimante)	libérer (table traçante)
libérer (table traçante)	libérer (modem)	libérer (imprimante)

On appelle p1, p2 et p3 les processus associés respectivement aux programmes 1, 2 et 3.

1. Les processus s'exécutent de manière concurrente.
   Justifier qu'une situation d'interblocage peut se produire.
2. Modifier l'ordre des instructions du programme 3 pour qu'une telle situation ne puisse pas se produire. Aucune justification n'est attendue.
3. Supposons que le processus p1 demande la table traçante alors qu'elle est en cours d'utilisation par le processus p3. Parmi les états suivants, quel sera l'état du processus p1 tant que la table traçante n'est pas disponible :

   a) élu b) bloqué c) prêt d) terminé

Partie B :

Avec une ligne de commande dans un terminal sous Linux, on obtient l'affichage suivant :

```
UID        PID  PPID  C STIME TTY          TIME CMD
...
pi        6211   831  8 09:07 ?        00:01:16 /usr/lib/chromium-browser/chromium-browser-v7 --disable-quic --enable-tcp-fast-open --p
pi        6252  6211  0 09:07 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi        6254  6252  0 09:07 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi        6294  6211  4 09:07 ?        00:00:40 /usr/lib/chromium-browser/chromium-browser-v7 --type=gpu-process --field-trial-handle=1
pi        6300  6211  1 09:07 ?        00:00:16 /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi        6467  6254  1 09:07 ?        00:00:11 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       11267  6254  2 09:12 ?        00:00:15 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       12035   836  0 09:13 ?        00:00:00 /usr/lib/libreoffice/program/oosplash --writer file:///home/pi/Desktop/mon_fichier.odt
pi       12073 12035  2 09:13 ?        00:00:15 /usr/lib/libreoffice/program/soffice.bin --writer file:///home/pi/Desktop/mon_fichier.o
pi       12253   831  1 09:13 ?        00:00:07 /usr/bin/python3 /usr/bin/sense_emu_gui
pi       20010  6211  1 09:21 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi       20029  6254 56 09:21 ?        00:00:28 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       20339  6254  4 09:21 ?        00:00:01 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       20343  6254  2 09:21 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       20464  6211 17 09:22 ?        00:00:00 /proc/self/exe --type=utility --field-trial-handle=1075863133478894917,6306120996223181
pi       20488  6254 14 09:22 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi       20519   676  0 09:22 pts/0    00:00:00 ps -ef
```

La documentation Linux donne la signification des différents champs :

- `UID` : identifiant utilisateur effectif ;
- `PID` : identifiant de processus ;
- `PPID` : PID du processus parent ;
- `C` : partie entière du pourcentage d'utilisation du processeur par rapport au temps de vie des processus ;
- `STIME` : l'heure de lancement du processus ;
- `TTY` : terminal de contrôle
- `TIME` : temps d'exécution
- `CMD` : nom de la commande du processus

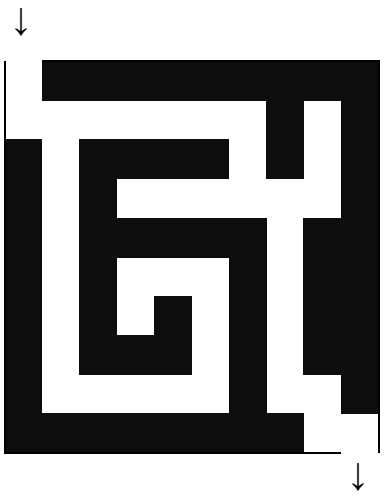
1. Parmi les quatre commandes suivantes, laquelle a permis cet affichage ?
   a) `ls -l`
   b) `ps –ef`
   c) `cd ..`
   d) `chmod 741 processus.txt`
2. Quel est l'identifiant du processus parent à l'origine de tous les processus concernant le navigateur Web (chromium-browser) ?
3. Quel est l'identifiant du processus dont le temps d'exécution est le plus long ?

Exercice 2

Thèmes abordés : programmation Python, tuples et listes

L'objectif de cet exercice est de mettre en place une modélisation d'un jeu de labyrinthe en langage Python.

On décide de représenter un labyrinthe par un tableau carré de taille `n`, dans lequel les cases seront des 0 si l'on peut s'y déplacer et des 1 s'il s'agit d'un mur. Voici un exemple de représentation d'un labyrinthe :



```
laby=[[0,1,1,1,1,1,1,1,1,1],
      [0,0,0,0,0,0,0,1,0,1],
      [1,0,1,1,1,1,0,1,0,1],
      [1,0,1,0,0,0,0,0,0,1],
      [1,0,1,1,1,1,1,0,1,1],
      [1,0,1,0,0,0,1,0,1,1],
      [1,0,1,0,1,0,1,0,1,1],
      [1,0,1,1,1,0,1,0,1,1],
      [1,0,0,0,0,0,1,0,0,1],
      [1,1,1,1,1,1,1,1,0,0]]
```

L'entrée du labyrinthe se situe à la première case du tableau (celle en haut à gauche) et la sortie du labyrinthe se trouve à la dernière case (celle en bas à droite).

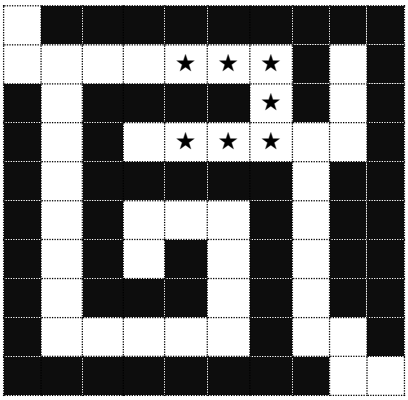
1. **Proposer**, en langage Python, une fonction `mur`, prenant en paramètre un tableau représentant un labyrinthe et deux entiers `i` et `j` compris entre 0 et `n-1` et qui renvoie un booléen indiquant la présence ou non d'un mur. Par exemple :

```
>>mur(laby, 2, 3)
True
>>mur(laby, 1, 8)
False
```

Un parcours dans le labyrinthe va être représenté par une liste de **cases.** Il s'agit de couples `(i,j)` où `i` et `j` correspondent respectivement aux numéros de ligne et de colonne des cases successivement visitées au long du parcours. Ainsi, la liste suivante

`[(1,4),(1,5),(1,6),(2,6),(3,6),(3,5),(3,4)]`

correspond au parcours repéré par des étoiles ⋆ ci-contre :



La liste `[(0,0),(1,0),(1,1),(5,1),(6,1)]` ne peut correspondre au parcours d'un labyrinthe car toutes les cases parcourues successivement ne sont pas adjacentes.

2. On considère la fonction `voisine` ci-dessous, écrite en langage Python, qui prend en paramètres deux cases données sous forme de couple.

```
def voisine(case1, case2) :
    l1, c1 = case1
    l2, c2 = case2
    # on vous rappelle que **2 signifie puissance 2
    d = (l1-l2)**2 + (c1-c2)**2
    return (d == 1)
```

2.a. Après avoir remarqué que les quantités `l1-l2` et `c1-c2` sont des entiers, **expliquer** pourquoi la fonction `voisine` indique si deux cases données sous forme de tuples `(l,c)` sont adjacentes.

2.b. **En déduire** une fonction `adjacentes` qui reçoit une liste de cases et renvoie un booléen indiquant si la liste des cases forme une chaîne de cases adjacentes.

Un parcours sera qualifié de ***compatible avec le labyrinthe*** lorsqu'il s'agit d'une succession de cases adjacentes accessibles (non murées). On donne la fonction `teste(cases, laby)` qui indique si le chemin `cases` est un chemin possible compatible avec le labyrinthe `laby` :

```
def teste(cases, laby) :
    if not adjacentes(cases) :
        return False
    possible = True
    i = 0
    while i < len(cases) and possible:
        if mur(laby, cases[i][0], cases[i][1]) :
            possible = False
        i = i + 1
    return possible
```

3. **Justifier** que la boucle de la fonction précédente se termine.
4. **En déduire** une fonction `echappe(cases, laby)` qui indique par un booléen si le chemin `cases` permet d'aller de l'entrée à la sortie du labyrinthe `laby`.