CONCOURS
CPGE
EPITA - IPSA - ESME
1 SEUL ET UNIQUE CONCOURS
POUR VOUS PRÉPARER
À UNE MULTITUDE
DE MÉTIERS

Samedi 17 Avril 2021

OPTION : SCIENCES DU NUMERIQUE

MP / PC / PSI / PT / TSI

Durée : 2 Heures

Condition(s) particulière(s)

Calculatrice interdite
Indiquer uniquement votre code candidat sur le QCM et l'insérer dans votre copie d'examen

Concours CPGE EPITA-IPSA-ESME 2021
Option Sciences du Numérique

Consignes Python

Tout code doit être écrit dans le langage Python.

- Tout code Python non indenté ne sera pas corrigé.
- Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué ci-dessous.
- Vous pouvez écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).
  Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.

Fonctions et méthodes autorisées

Vous pouvez utiliser la méthode `append` et la fonction `len` ainsi que la fonction `range` :

```
>>> L = []

>>> for i in range(5):
        L.append(i)

>>> L
[0, 1, 2, 3, 4]

>>> len (L)
5

>>> for i in range(5, 10):
        L.append(i)

>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Aucun opérateur n'est autorisé sur les listes (+, *, ==, …).

Introduction

Le sujet de l'option informatique contient cinq exercices allant de l'expression de principes algorithmiques jusqu'à l'écriture de fonctions. Le langage utilisé, dans tous les cas, est Python. Les deux premiers exercices portent sur l'aspect théorique de problèmes et les trois derniers sur le codage de fonctions sur les listes. Vous trouverez en annexe les différentes fonctions et structures de données autorisées.

A) Double garage …

Imaginons un garage possédant deux voies d'entrée et une seule de sortie (voir Figure 1). Pour garer sa voiture il existe 2 voies d'accès (*entrée e1* et *entrée e2*) et pour la ressortir une seule voie (*sortie s*).

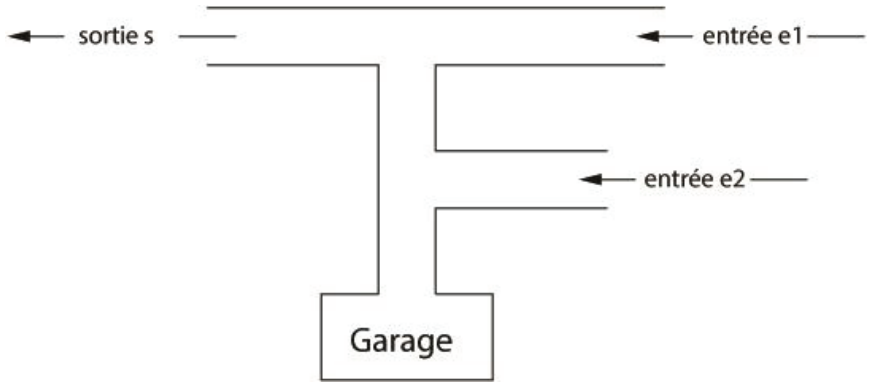

Figure 1 - Garage à deux entrées et une sortie

Les mouvements possibles des voitures sont :

- une voiture peut entrer par l'*entrée e1*
- une voiture peut entrer par l'*entrée e2*
- une voiture ne peut sortir que par la *sortie s*
- une voiture ne peut ressortir que si elle est la dernière entrée (peu importe l'entrée)
- une voiture ne peut pas sauter par-dessus une autre

Remarques :

- *Il n'y a pas de problème de voitures se retrouvant face à face.*
- *Le garage est vide au départ et doit l'être à la fin.*

Symbolisons une entrée à l'aide du couple ($v_{\text{numéro du véhicule}}$, $e_{\text{numéro de l'entrée empruntée}}$) et la sortie simplement par la lettre s.

1. Est-ce que les séquences d'entrées/sorties suivantes de véhicules sont valides ?

   a) $(v_1,e_1),(v_2,e_1),(v_3,e_1),s,s,(v_4,e_2),(v_5,e_1),s,s,s,(v_6,e_2),s$

   b) $(v_1,e_1),(v_2,e_2),s,(v_3,e_2),s,s,s,(v_4,e_1),(v_5,e_2),s,(v_6,e_1),(v_7,e_2),s,s$

   Dans les cas où la séquence n'est pas valide, indiquer brièvement pourquoi.

2. Donner une règle qui caractériserait les séquences admissibles.

B) Dichotomie …

On considèrera une version de l'algorithme de recherche dichotomique, d'un élément dans une liste triée, qui s'arrête dès que les bornes de recherche se croisent ou sont identiques.

1. Dessiner l'arbre de décision de la recherche dichotomique dans une liste de 16 éléments. Chaque nœud représente l'intervalle de recherche (les bornes gauche et droite) ainsi que l'indice calculé du milieu (voir figure 2).

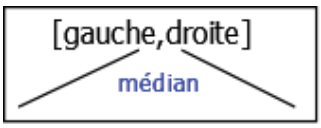

Figure 2 – Nœud de l'arbre de décision

2. Soit une liste de 32768 éléments triés en ordre croissant. Combien de comparaisons d'éléments seront faites, au pire des cas, dans le cas d'une recherche négative (réponse entière) ?
3. Soit k la réponse à la question précédente. Quelle peut-être, au maximum, la longueur de la liste qui générera k + 2 comparaisons lors d'une recherche négative ?

C) Recherche des indices …

Écrire la fonction **search_indexes(L, val)** qui retourne les valeurs du premier et du dernier indices de l'élément **val** dans la liste triée en ordre croissant **L**.
Si l'élément **val** n'est pas dans la liste, la fonction retourne le couple **(-1, -1)**.

Exemples d'applications :

```
>>> search_indexes([1, 4, 12, 12, 42], 4)
(1, 1)

>>> search_indexes([1, 4, 12, 12, 42], 15)
(-1, -1)

>>> search_indexes([1, 4, 12, 12, 42], 12)
(2, 3)

>>> search_indexes([4, 4, 4], 4)
(0, 2)
```