L'attention du détail
Problem ID: stringdiff

Niveau 1

Ton ami a du mal à trouver voir les différences entre les phrases suivantes :

- "je suis une longue phrase qui n'a aucun sens"
- "je suis une longue phrose qii n'a aucun senz"

Il te dit que ça lui arrive souvent, tu décides de crées un algorithme pour lister tout les endroit de la phrase où les deux phrases sont différentes sous la forme : (position de différence, lettre de la phrase 1 en cette position, lettre de la phrase 2 en cette position)

Input

Deux lignes de chaines de caractères à comparer.

Output

Une liste de tuple contenant en première position la position de l'erreur, en deuxième position la lettre de la phrase 1 en cette position et en dernière position la lettre de la phrase 2 en cette position.

Si les deux chaines de caractères sont identiques, alors affichez *Phrases identiques*.

Code de démarrage

Pour plus de facilité, vous pouvez copier-coller ce code afin de ne vous concentrer que sur le code à proprement parler.

```
string1 = input()
string2 = input()

def stringDiff(s1,s2):
    pass

liste = stringDiff(string1,string2)

for t in liste:
    print(' '.join(map(str,i)))
```

Sample Input 1	**Sample Output 1**
je suis une longue phrase qui n'a aucun sens je suis une longue phrose qii n'a aucun senz	22 a o 27 u i 43 s z

Sample Input 2	**Sample Output 2**
CPUMons c'est trop bien ! CPUMons c'est trop bien !	Phrases identiques

2 Exercice 2

Le Kot associatif CPUMons a besoin d'aide !

La liste des membres commence à s'allonger et il serait judicieux d'établir un système de gestion de membre avant que cette liste devienne ingérable. Pour cela nous avons besoin :

- D'un objet membre qui a comme caractéristiques : nom, prénom, adresse email umons, faculté, section, date d'inscription (sous la forme d'un nombre aammjj), son nombre de participation, ainsi que les dates auxquelles il a participé à un entraînement (aammjj).
- L'adresse email doit être construite sur base du prénom et du nom (prénom.nom@student.umons.ac.be).
- On a besoin de connaître quels membres étaient présents à une date donnée ainsi que le nombre de membres présents à cette séance.
- Pour établir un graphique du nombre de membres présents à chaque date, on a besoin de connaître le nombre d'étudiants pour toutes les séances.
- Les membres seront stockés dans un tableau.
- On doit pouvoir ajouter, supprimer des membres du tableau.

BONUS : Ouuups on s'est trompé dans nos demandes ! Si on a trop de membres, notre petit serveur ne pourra pas gérer ce tableau. Stockez les membres dans une liste chaînée pour plus d'efficacité !

Un problème de tasses

Problem ID: tasses

Niveau 3

Dans un tableau, une tasse est une séquence d'éléments consécutifs (allant de i à j avec i+2 $\leqslant$ j) telle que :

$$A[i] > A[i+1] = ... = A[j-1] < A[j].$$

La longueur d'une tasse est le nombre d'éléments égaux qu'elle contient. Notez qu'une tasse contient au moins 3 éléments et que la plus petite longueur pour une tasse est donc 1.

En vous basant sur la technique des invariants, concevez un algorithme qui retourne la longueur de la plus grande tasse d'un tableau A de $n \geqslant 1$ entiers. (Si A ne contient pas de tasse, l'algorithme retourne 0.)

Input

Un tableau T non-vides d'entiers.

Output

Un entier représentant la longueur de la plus grande tasse.

Challenge

Refaire le code mais avec une complexité linéaire (c'est à dire en O(n))!

Code de démarrage

Pour plus de facilité, vous pouvez copier-coller ce code afin de ne vous concentrer que sur le code à proprement parler.

```
tab = list(map(int, input().split()))

def nombre_tasses(tab):
    pass

print(nombre_tasses(tab))
```

Sample Input 1	**Sample Output 1**
1 2 3 2 2 1 2 2 6	1

Sample Input 2	**Sample Output 2**
9 9 5 5 5 6 4 4 3 3 3 3 8	4

Le puzzle perdu

Problem ID: puzzle

Niveau 1

La Confédération des Puzzles Ultimement et Monstrueusement Ordinaires des Nationalistes Suédois à besoin de vous. En effet, la confédération a retrouvé une grosse boite remplie de pièces de puzzle mais ne savent pas si le puzzle est complet ou pas ou si la boite contient de multiples puzzles. Pour les aider, vous allez concevoir un algo qui détermine la taille du puzzle quand celui-ci est complet(car oui malheureusement la taille du puzzle est effacée). Pour vous donner un coup de main, les confédérés ont déjà trié les pièces en 3 catégories :

- Les pièces de coins qui touchent 2 bords du puzzles
- Les pièces de bords qui touchent 1 bord du puzzles
- Les pièces centrales qui ne touchent pas de bord

Input

Une ligne contenant 3 entiers c,b,m ($0 \leq c, b, m \leq 10^9$), respectivement les pièces de coins, bords et milieu.

Output

Soit w,h la longueur et hauteur d'un puzzle. S'il existe des nombres w et h satisfaisant $w \geq h \geq 2$ tels que la taille originale du puzzle aurait pu être w×h, alors sortez une seule ligne contenant w et h. Sinon, sortez "impossible".

Sample Input 1	**Sample Output 1**
`4 4 1`	`3 3`

Sample Input 2	**Sample Output 2**
`0 2 4`	`impossible`

Exercice 5

Le batsignal résonne sur la ville. Batman doit rechercher les otages d'un bâtiment donné en sautant de fenêtre en fenêtre à l'aide de son grappin. Le but de Batman est d'arriver sur la fenêtre de la pièce où les otages se trouvent afin de désamorcer les bombes du Joker. Malheureusement il n'a qu'un nombre de sauts limités avant que les bombes n'explosent et à donc besoin de vous petits codeurs de l'ombre.

Règles : Avant chaque saut, le détecteur fournira à Batman la direction des bombes par rapport à la position actuelle de Batman :

- U (Up : les bombes sont situées au dessus de Batman)
- UR (Up-Right : les bombes sont situées au dessus et à droite de Batman)
- R (Right : les bombes sont situées à droite de Batman)
- DR (Down-Right : les bombes sont situées en dessous et à droite de Batman)
- D (Down : les bombes sont situées en dessous de Batman)
- DL (Down-Left : les bombes sont situées en dessous et à gauche de Batman)
- L (Left : les bombes sont situées à gauche de Batman)
- UL (Up-Left : les bombes sont situées au dessus et à gauche de Batman)

Votre mission consiste à programmer le détecteur afin qu'il indique la position de la fenêtre sur laquelle Batman devra se rendre au saut suivant de sorte qu'il atteigne les bombes le plus tôt possible.
Les bâtiments sont représentés par des rectangles de fenêtres, la fenêtre en haut à gauche a pour position (0,0).

Entrée de jeu : Le programme doit d'abord lire les données d'initialisation depuis l'entrée standard, puis, dans une boucle infinie, lire depuis l'entrée standard les données relatives à l'état courant de Batman et fournir sur la sortie standard les données demandées.

Entrée : Ligne 1 : 2 entiers W H. Le couple (W, H) représente la largeur et la hauteur du bâtiment en nombre de fenêtres.
Ligne 2 : 1 entier N, qui représente le nombre de sauts que Batman peut faire avant que les bombes n'explosent.
Ligne 3 : 2 entiers X0 Y0, qui représentent la position de départ de Batman.

Entrée pour un tour de jeu : La direction vers laquelle se trouve la bombe.

Sortie pour un tour de jeu : Une ligne unique avec 2 entiers X Y séparés par un espace. (X, Y) représente la position de la prochaine fenêtre sur laquelle Batman devrait sauter. X représente l'index sur l'axe horizontal, Y représente l'index sur l'axe vertical. (0,0) se trouve dans le coin haut gauche du bâtiment.

indice : Pour un indice : Cliquez ici (attention besoin du son de la vidéo)

Le Président
Problem ID: president

Niveau 2

Dans le jeu de cartes "Président", on dispose d'un paquet de 52 cartes classique. Le principe est de toujours mettre une carte plus forte que la précédente. Par exemple, si je met un 10, la personne peut mettre : un 10, un Valet, une Dame, un Roi ou un As. La personne a également le droit de décider de "couper" en plaçant un 2. Elle peut alors rejouer en mettant la carte de son choix. On comprends alors que l'As est meilleure que le Roi et le 2 est la carte la plus forte dans ce jeu.

Créez une fonction qui trie un deck de cartes en fonction de ces valeurs.

Input

Une chaine de caractères séparé par des espaces qui représente le jeu de carte.

Output

Une chaine de caractères séparé par des espaces du jeu de carte trié.

Code de démarrage

Pour plus de facilité, vous pouvez copier-coller ce code afin de ne vous concentrer que sur le code à proprement parler.

```
def sort_deck(deck):
    pass

deck = input().split()
print("␣".join(sort_deck(deck)))
```

Sample Input 1	**Sample Output 1**
A 2 3 K 10	3 10 K A 2

Sample Input 2	**Sample Output 2**
A A 2 4 6 5 9 5	4 5 5 6 9 A A 2

Indenter son code
Problem ID: correctIdentation

Niveau 2

La syntaxe du langage C est en elle-même récursive. Ainsi, un bloc d'instructions encadré par des accolades '' et '', peut lui-même contenir un bloc d'instructions, encadré par des accolades, etc. Pour bien mettre en évidence les différents niveaux d'imbrications d'un programme, une bonne habitude consiste à indenter son code, c'est à dire à ajouter plus ou moins d'espaces devant chaque ligne, suivant le niveau d'imbrication du bloc qui la contient. De plus, il peut être une bonne idée de mettre chaque accolade seule sur sa ligne, pour bien la mettre en évidence, et avoir un code aéré.

Dans cet exercice, vous allez écrire un programme qui s'occupe d'indenter automatiquement des blocs d'instructions. On ne vous demande cependant pas de travailler sur un programme C complet, mais simplement sur les accolades.

On vous donne une ligne de texte, composée uniquement d'accolades ouvrantes et fermantes, sans espaces. Les accolades sont structurées correctement, comme des blocs d'un programme C valide.

Vous devez afficher ces accolades en les plaçant une par ligne, et en les indentant. Vous devez placer devant chaque accolade, 3 espaces par niveau d'imbrication de cette accolade.

Votre programme doit impérativement être basé sur une fonction récursive, et non sur une boucle..

Input

L'entrée contient une chaîne composée d'accolades ouvrantes et fermantes, sans espaces, suivie d'un retour à la ligne.

Output

N lignes contenant une accolade dans le bon sens bien indenté;e

Constraints

- $2 \leq N \leq 1000$, où N est le nombre de caractères (accolades ouvrantes ou fermantes) de la chaîne.

Code de démarrage

Pour plus de facilité, vous pouvez copier-coller ce code afin de ne vous concentrer que sur le code à proprement parler.

```
text = input()
```

Sample Input 1	Sample Output 1

{{{}}}	{
	    {
	        {
	        }
	    }
	}

Sample Input 2	Sample Output 2

{{}{{}}}	{
	{
	}
	{
	{
	}
	}
	}