ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵓⵙⴳⵎⵉ ⴷ ⵜⵣⵔⴼⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ
ⴰⵙⵙⵏⵉⵎⵔ ⵏ ⵜⵙⴷⴰⵡⵉⵜ ⴰⵙⵙⵏⵉⵎⵔ ⵏ ⵜⵣⵔⴼⵜ

المملكة المغربية
وزارة التربية الوطنية والتكوين المهني
والتعليم العالي والبحث العلمي

Royaume du Maroc

Ministère de l'Éducation Nationale, de la Formation Professionnelle, de l'Enseignement Supérieur et de la Recherche Scientifique

Département de l'Enseignement Supérieur et de la Recherche Scientifique

وزارة التجهيز و النقل و اللوجستيك و الماء
Ministère de l'Equipement, du Transport, de la Logistique et de l'Eau

المدرسة الحسانية للأشغال العمومية
École Hassania des Travaux Publics

CNC 2021

Concours National Commun

d'Admission dans les Établissements de Formation d'Ingénieurs et Établissements Assimilés

Épreuve d'**Informatique**

Filière : **PSI**

Durée **2 heures**

Cette épreuve comporte **10 pages au format A4**, en plus de la page de garde

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- Toutes les instructions et les fonctions demandées seront écrites en Python ;
- Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈≈

Important : Le candidat doit impérativement traiter l'exercice ci-dessous, et inclure les réponses dans les premières pages du cahier de réponse.

Exercice : (4 points)

Polynôme et liste

Un polynôme à coefficients réels peut être représenté par une liste ***P*** telle que : chaque élément ***P[i]*** contient le coefficient du monôme de degré ***i***.

Exemples :

- ✓ Le polynôme $\mathbf{4 + x - 5x^2 + 3x^4}$ est représenté par la liste **[4, 1, -5, 0, 3]**
- ✓ La liste **[-1, 4, 0, 7, 0, 1, -2]** représente le polynôme $\mathbf{-1 + 4x + 7x^3 + x^5 - 2x^6}$

0.5 pt **Q.1-** Écrire la liste qui représente le polynôme : $\mathbf{x^5 - 3x^4 + 7x^2 + 2}$

1.25 pt **Q.2-** Écrire la fonction ***produit (k, P)***, qui reçoit en paramètres un réel ***k*** non nul et une liste ***P*** qui représente un polynôme. La fonction retourne la liste qui représente le polynôme ***k*P***.

Exemple :

La fonction **produit (5, [8, 1, -2, 0, 3])** retourne la liste **[40, 5, -10, 0, 15]**

0.75 pt **Q.3-** Déterminer la complexité de la fonction **produit (k, P)**. Justifier votre réponse.

1.5 pt **Q.4-** Écrire la fonction ***image (P, x)***, qui reçoit en paramètres une liste ***P*** qui représente un polynôme, et un réel ***x***. La fonction retourne la valeur de **P(x)** image de ***x*** par le polynôme ***P***.

Exemple :

La fonction **image ([8, 1, -5, 0, 3], 3)** retourne le nombre $\mathbf{209} = 8*3^0 + 1*3^1 - 5*3^2 + 0*3^3 + 3*3^4$

Partie I : Calcul numérique

Modélisation d'une épidémie

Le modèle 'SIR'

La crise sanitaire mondiale du Coronavirus Covid-19 a démontré le rôle des modélisations mathématiques dans la prise de décisions politiques et sanitaires.

Les modèles à compartiments font partie des premiers modèles mathématiques, à avoir été utilisés en épidémiologie. L'idée est de diviser une population en plusieurs compartiments (groupes d'individus), et définir les règles d'échanges entre ces compartiments.

Le modèle SIR

Dans ce modèle, les individus sont répartis en trois compartiments :

- **S**, dans lequel se trouvent les individus sains et susceptibles d'être infectés ;
- **I**, pour les individus infectés et infectieux ;
- **R**, pour les individus retirés du modèle, non susceptibles d'être infectés, car guéris et immunisés ou décédés.

Le modèle **SIR** décrit l'évolution dans le temps du nombre d'individus dans chaque compartiment, et part du schéma simplifié suivant :

- β : représente le taux de transmission de la maladie d'un individu infecté à un individu sain ;
- γ : représente le taux de guérison.

Un individu est d'abord sain (**S**) ; il est infecté (**I**) avec la probabilité (β) ; enfin, il guérit (**R**), avec la probabilité (γ).

Il paraît plus naturel de travailler avec le nombre des individus dans chaque catégorie, mais certains calculs seront plus simples si on utilise plutôt la proportion des individus dans chaque catégorie, ce qui nous permet de connaître tout aussi bien la progression de l'épidémie.

On note donc $S(t)$, $I(t)$ et $R(t)$ les proportions des individus dans chaque catégorie à l'instant t.

L'évolution des trois catégories de population peut alors être décrite par le système d'équations différentielles suivant :

$$(E)\quad \begin{cases} S'(t) = -\beta.S(t).I(t) & (1) \\ I'(t) = \beta.S(t).I(t) - \gamma.I(t) & (2) \\ R'(t) = \gamma.I(t) & (3) \end{cases}$$

Les dérivées permettent de connaître la variation (c'est à dire si c'est croissant ou décroissant) des fonctions $S(t)$, $I(t)$ et $R(t)$ en fonction du temps t, afin d'en décrire l'évolution au cours du temps.

Le produit $S(t).I(t)$ représente le nombre de contacts entre des individus sains et des individus infectés. β étant le taux de transmission, il y a dès lors $\beta.S(t).I(t)$ individus nouvellement infectés. Ceux-ci se soustraient des individus sains (équation 1), et s'ajoutent aux individus infectés (équation 2).

De même, parmi les individus infectés, certains vont guérir. γ étant le taux de guérison, il a $\gamma.I(t)$ individus nouvellement guéris qui se soustraient des individus infectés (équation 2) et s'ajoutent aux individus retirés (équation 3).

On suppose que les modules **numpy** et **matplotlib.pyplot** sont importés :

```
import numpy as np
import matplotlib.pyplot as pl
```

Méthodes de Runge-Kutta :

Dans cette partie, nous allons nous intéresser à une méthode d'ordre **2**, appelée **Runge-Kutta 2** (RK2) permettant de résoudre numériquement les équations différentielles du premier ordre, avec une condition initiale, sous la forme :

$$\begin{cases} y' = f(t, y(t)) \quad ; \quad \forall\, t \in [t_0, t_0 + \mathbf{T}] \\ y(t_0) = y_0 \end{cases}$$

On note $\mathbf{I} = [t_0, t_0 + \mathbf{T}]$ l'intervalle de résolution. Pour un nombre de nœuds **N** donné, soit $t_n = t_0 + n*h$, avec $n = 0, 1, 2, ..., \mathbf{N}$, une suite de nœuds de **I** induisant une discrétisation de **I** en sous-intervalles $\mathbf{I_n} = [t_n , t_{n+1}]$. La longueur h de ces sous-intervalles est appelée **pas** de discrétisation. Le pas de discrétisation h est donné par $h = \frac{T}{N}$.

Soit y_n l'approximation au nœud t_n de la solution exacte $y(t_n)$. Le schéma de la méthode **RK2** est donné par :

- y_0 est donné
- $y_{n+1} = y_n + \frac{h}{2} * (f(t_n, y_n) + f(t_n + h, k))$

  avec : $k = y_n + h * f(t_n, y_n)$

Q.1- Écrire la fonction **def RK2 (f, t_0, T, y_0, N):** qui reçoit en paramètres la fonction f: (t , y) → f (t , y), t_0 et **T** tels que $[t_0 , t_0 + T]$ est l'intervalle de résolution, la condition initiale y_0 et **N** est le nombre de nœuds. La fonction retourne **tt** = $[t_0 , t_1 , ... , t_N]$ et **Y** = $[y_0 , y_1 , ... , y_N]$ en utilisant le schéma de RK2.

Q.2- Le but de cette question est d'utiliser la fonction précédente **RK2**, pour résoudre numériquement le système d'équations différentielles (**E**).

Q.2.a- En posant **Y(t) = [S(t) , I(t) , R(t)]**, montrer que le système d'équations différentielles (**E**) peut s'écrire sous la forme : **Y'(t)=F(t, Y(t))** avec

$$\mathbf{F} \quad : \quad \mathbf{I} * \mathbf{IR}^3 \quad \rightarrow \quad \mathbf{IR}^3$$

$$(t , X = [\, X[0] , X[1] , X[2] \,]) \quad \rightarrow \quad [\, z_0 , z_1 , z_2 \,]$$

z_0 , z_1 et z_2 sont à préciser en fonction de **X[0], X[1]** et **X[2]**.

Q.3- On suppose que **beta** et **gamma** sont deux variables globales, initialisées par les valeurs suivantes :

beta = 0.5

gamma = 0.1

En utilisant la fonction précédente **RK2**, écrire un script python qui permet de tracer la représentation graphique ci-dessous (Figure 1), des fonctions **S(t)**, **I(t)** et **R(t)**. Le nombre de points générés dans chaque courbe est $\mathbf{N=10^3}$.

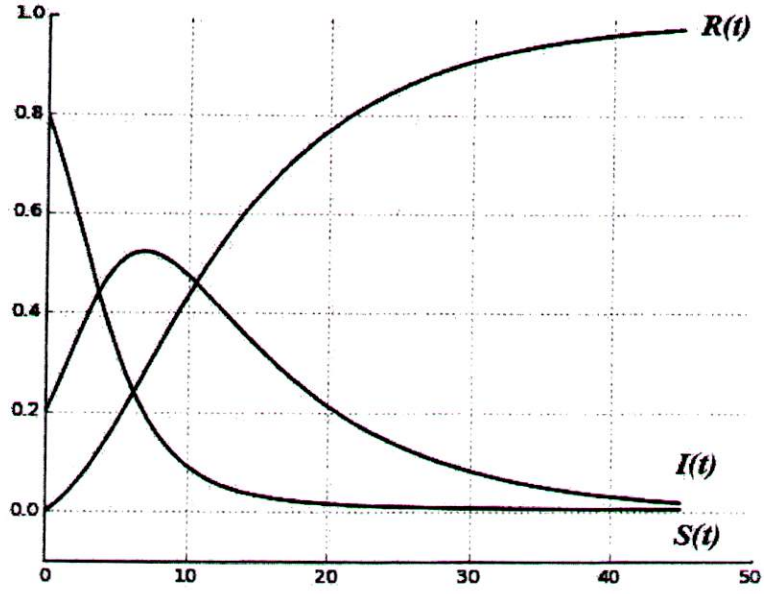

Figure 1

Partie II : Bases de données et langage SQL

Dans le but de suivre l'évolution de la pandémie du Covid-19 dans chaque pays, on propose d'utiliser une base de données composée de **3** tables : **Pays, Villes** et **Quotidien**.

a- Structure de la table 'Pays' :

La table **'Pays'** est composée de trois champs : Le champ '**codPays**' de type texte. Ce champ représente la clé primaire de la table 'Pays'. Le champ '**nom**' de type texte, contient le nom de chaque pays.

Exemples :

codPays	nom
DEU	Allemagne
FRA	France
ESP	Espagne
ARN	Suède
ITA	Italie
...	...

b- Structure de la table 'Villes' :

La table **'Villes'** est composée de 4 champs : Le champ '**codVille**' de type texte. Il représente la clé primaire de la table 'Ville'. Le champ '**nom**' de type texte, il contient le nom de chaque ville. Le champ '**population**' de type entier, il contient le nombre de personnes dans chaque ville. Le champ '**codP**' de type texte, il contient le code du pays de chaque ville.

Exemples :

codVille	nom	population	codP
CIA	Rome	2 844 395	ITA
NAP	Napoli	954 244	ITA
PAR	Paris	2 148 271	FRA
TXL	Berlin	3 748 148	DEU
LYS	Lyon	1 669 730	FRA
	...	...	...

c- Structure de la table 'Quotidien' :

La table 'Quotidien' est composée de 5 champs : Le champ '**jour**' de type date, il contient la date de chaque jour. Le champ '**infectés**' de type entier, il contient le nombre de nouveaux cas de personnes infectées par le virus, chaque jour dans chaque ville. Le champ '**rétablis**' de type entier, il contient le nombre de nouveaux cas de personnes guéries du virus, chaque jour dans chaque ville. Le champ '**décédés**' de type entier, il contient le nombre de nouveaux cas de personnes décédées à cause du virus, chaque jour dans chaque ville. Le champ '**codV**' de type texte, il contient les codes des villes.

Exemples :

jour	infectés	rétablis	décédés	codeV
2020-07-15	4653	3458	172	CIA
2020-07-15	6074	4892	493	TXL
2020-07-15	1762	2059	651	LYS
2020-07-16	3721	5973	85	CIA
...	...			...

Q.1- Le jour du '**17/09/2020**', la ville de code '**TXL**' a enregistré **2513** nouveaux cas de personnes infectées, **2847** nouveaux cas de personnes rétablies, et **65** nouveaux cas de personnes décédées.

Écrire, en langage **SQL**, la requête qui ajoute cet enregistrement dans la table 'Quotidien'.

Q.2- Écrire, en langage **SQL**, la requête qui donne pour résultat, le compte des villes qui ont enregistré des nombres de nouveaux cas de personnes infectées, composés d'au moins **5** chiffres, et ce le jour du **'27/08/2020'**.

Q.3- Écrire, en langage **SQL**, la requête qui donne pour résultat, le nombre de nouveaux cas de personnes infectées, le nombre nouveaux cas de personnes rétablies et le nombre de nouveaux cas de personnes décédées, chaque jour du mois de **Septembre 2020**, dans toutes les villes du pays '**Italie**'.

Q.4- Écrire, en algèbre relationnelle, la requête de la question **Q.3**

Q.5- *Le **taux d'incidence TI** pour une ville est égal au nombre de nouveaux cas de personnes infectées dans cette ville, pendant une période donnée, divisé par la population de cette ville. Le **TI** s'exprime en nombre de personnes pour **100000** personnes.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les codes des pays, les noms des villes et le **TI** de chaque ville, pendant l'année **2020**, tels que **TI** est compris entre **500** et **1000**, triés dans l'ordre croissant du **TI**.

Q.6- *Le **taux de guérison TG** pour un pays est égal au nombre total de personnes rétablies dans ce pays, divisé par le nombre total de personnes infectées dans ce même pays. Le **TG** s'exprime en pourcentage.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les noms des pays et le **TG** pour chaque pays, tels que **TG > 70%**, triés dans l'ordre décroissant du **TG**.

Q.7- *Le **taux de mortalité TM** pour une ville est égal au nombre total de personnes décédées dans cette ville, divisé par le nombre total de personnes infectées dans cette même ville. Le **TM** s'exprime en pourcentage.*

Écrire, en langage **SQL**, la requête qui donne pour résultat, les noms des pays, le plus petit **TM**, le plus grand **TM** et la moyenne des **TM** des villes dans chaque pays, triés dans l'ordre croissant du nombre de villes dans chaque pays.

Partie III : Problème Python

Distance de 'Levenshtein'

En mathématiques et en informatique théorique, une **mesure de similarité**, plus exactement une mesure de **distance entre deux chaînes de caractères**, est une façon de représenter par un nombre la différence entre deux chaînes de caractères. Cela permet de comparer deux chaines de façon simple et pratique. C'est donc une forme de distance mathématique et de métrique pour les chaînes de caractères.

En programmation, la mesure la plus simple et la plus courante est la **distance de Levenshtein**, aussi connue sous le terme *distance d'édition*. De telles distances ont été étendues pour s'appliquer également à la comparaison phonétique, à l'analyse lexicale, aux comparaisons grammaticales et autres…

La distance de **Levenshtein** entre deux chaînes **A** et **B** est le nombre minimal d'opérations élémentaires pour transformer **A** en **B**, en effectuant seulement les opérations élémentaires suivantes :

- ✓ **suppression** (ou effacement) d'un caractère de la chaîne **A** ;
- ✓ **insertion** (ou ajout) dans la chaîne **A** d'un caractère de la chaîne **B** ;
- ✓ **substitution** (ou remplacement) d'un caractère de la chaîne **A** par un caractère différent, de la chaîne **B**.

Q.1- Écrire la fonction ***supprimer (S, k)*** qui reçoit en paramètres une chaîne de caractères **S** non vide et un entier positif **k** qui représente l'indice d'un caractère dans **S**. La fonction retourne la chaîne qui contient tous les caractères de **S**, sauf le caractère d'indice **k** dans **S**.

Exemples :

- ✓ La fonction **supprimer ('vendredi', 3)** retourne la chaîne : **'venredi'**
- ✓ La fonction **supprimer ('vendredi', 0)** retourne la chaîne : **'endredi'**

Q.2- Écrire la fonction ***inserer (S, c, k)*** qui reçoit en paramètres une chaîne de caractères **S** non vide, un caractère **c** et un entier positif **k** tel que $0 \leq k \leq$ **taille(S)**. La fonction retourne la chaîne de caractères qui contient tous les caractères de **S**, en insérant le caractère **c** à la position d'indice **k**.

Exemples :

- ✓ La fonction **inserer ('vedredi', 'n', 2)** retourne la chaîne : **'vendredi'**
- ✓ La fonction **inserer ('samed', 'i', 5)** retourne la chaîne : **'samedi'**

Q.3- Écrire la fonction ***remplacer (S, c, k)*** qui reçoit en paramètres une chaîne de caractères **S** non vide, un caractère **c** et un entier positif **k** qui représente l'indice d'un caractère dans **S**. La fonction retourne la chaîne de caractères qui contient les caractères de **S**, en remplaçant le caractère d'indice **k** par le caractère **c**.

Exemple :

- ✓ La fonction **remplacer ('mendredi', 'v', 0)** retourne la chaîne : **'vendredi'**
- ✓ La fonction **remplacer ('saredi', 'm', 2)** retourne la chaîne : **'samedi'**

Algorithme de 'Wagner et Fisher' :

L'algorithme, dû à Wagner et Fischer (1974), permet de calculer la distance de Levenshtein entre deux chaînes de caractères. Cet algorithme est un exemple de programmation dynamique (solution de type du bas en haut), qui utilise une matrice.

Pour calculer la distance de Levenshtein entre deux chaînes **A** et **B**, l'algorithme de Wagner et Fischer effectue **3** étapes :

Étape 1 :

- **a** et **b** sont respectivement les longueurs des deux chaînes **A** et **B** ;
- On crée une matrice **C** de **a+1** lignes et **b+1** colonnes, remplie par des **0** ;
- On initialise la première ligne de **C** par les entiers : **0, 1, 2, 3, …, a**
- On initialise la première colonne de **C** par les entiers : **0, 1, 2, 3, …, b**

Étape 2 :

On associe à chacune des opérations (suppression, insertion ou substitution) un coût égal à **1**. Et on calcule les éléments $C_{i,j}$ tels que **i>1** et **j>1**, selon la règle suivante :

- Si A_{i-1} et B_{j-1} sont différents, alors $C_{i,j}$ est égale au minimum des **3** nombres suivants :
  $1 + C_{i-1,j}$, $1 + C_{i,j-1}$ et $1 + C_{i-1,j-1}$
- Sinon, $C_{i,j}$ est égale au minimum des **3** nombres suivants : $1 + C_{i-1,j}$, $1 + C_{i,j-1}$ et $C_{i-1,j-1}$
  NB : $1 + C_{i-1,j}$ correspond à une suppression dans **A**, $1 + C_{i,j-1}$ correspond à une insertion dans **A**, et $1 + C_{i-1,j-1}$ correspond à une substitution dans **A**.

Étape 3 :

Après avoir calculer tous les éléments de la matrice **C**, la valeur de l'élément $C_{a,b}$ est la valeur de la distance de Levenshtein entre les deux chaînes **A** et **B**.

Par définition d'une distance au sens mathématique du terme, selon cet algorithme, **0** est la distance de Levenshtein entre deux chaînes de caractères égales.

Q.4- Écrire la fonction ***etape1 (A, B)*** qui reçoit en paramètres deux chaînes de caractères **A** et **B**. La fonction réalise l'étape **1** de l'algorithme de Wagner et Fisher, et elle retourne une liste de listes qui représente la matrice **C**.

Exemple :

La fonction ***etape1 ('samedi', 'vendredi')*** retourne la liste de listes **C**, qui représente la matrice de **7** lignes et **9** colonnes :

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

Q.5- Écrire la fonction ***etape2 (A, B, C)*** qui reçoit en paramètres deux chaînes de caractères **A** et **B**, et la matrice **C**, résultat de l'étape **1**. La fonction effectue l'étape **2** de l'algorithme de Wagner et Fisher.

Exemple :

Après l'appel de la fonction ***etape2 ('samedi', 'vendredi', C)***, la matrice **C** devient ainsi :

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

Q.6- déterminer la complexité de la fonction ***etape2 (A, B, C)***, et justifier votre réponse.

Q.7- Écrire la fonction ***WF (A, B)*** qui reçoit en paramètres deux chaînes de caractères **A** et **B**, et qui retourne la valeur de la distance de levenshtein entre **A** et **B**, selon l'algorithme de Wagner et Fisher.

Exemples :

- La fonction ***WF ('samedi', 'vendredi')*** retourne le nombre **5**
- La fonction ***WF ('samedi', 'samedi')*** retourne le nombre **0**
- La fonction ***WF ('samedi', 'lundi')*** retourne le nombre **4**

Une des applications de la distance de Levenshtein est la correction orthographique : lorsqu'une personne tape un mot **S**, on le compare aux mots d'une liste **D**. Si le mot **S** n'est pas présent dans la liste **D**, alors on cherche parmi les mots de la liste **D** ceux dont la distance de Levenshtein à **S** est inférieure à une limite donnée. Comme remplacement à **S**, les mots de **D** les plus proches à **S** sont suggérés, dans l'ordre croissant de la distance de Levenshtein.

Q.8- Écrire la fonction ***tri_Levenshtein (D, S)*** qui reçoit en paramètres une liste **D** de chaînes de caractères non vides, et une chaîne de caractères **S** non vide. La fonction trie les éléments de **D** dans l'ordre croissant de la distance de levenshtein entre **S** et chaque élément de **D**.

Exemple :

D = ['lundi' , 'mardi' , 'mercredi' , 'jeudi' , 'vendredi' , 'dimanche']

Après l'appel de la fonction ***tri_ Levenshtein (D, 'samedi')***, on obtient la liste :

['mardi' , 'jeudi' , 'lundi' , 'vendredi' , 'mercredi' , 'dimanche']

Q.9- Écrire la fonction ***correction_orthographe (S, D, k)*** qui reçoit en paramètres une chaîne de caractères **S** non vide, une liste **D** de chaînes de caractères non vides, et un entier positif **k**. La fonction retourne la liste **L** des éléments de **D** tels que :

- La distance de Levenshtein entre **S** et chaque élément de **L** est inférieure ou égal à **k** ;
- Les éléments de **L** sont triés dans l'ordre croissant de la distance de Levenshtein entre **S** et chaque élément de **L**.

Exemple :

D = ['lundi' , 'mardi' , 'mercredi' , 'jeudi' , 'vendredi' , 'samedi' , 'dimanche']

La fonction ***correction_orthographe ('merdredi', D, 4)*** retourne la liste : ['mercredi' , 'vendredi' , 'mardi']

Séquence des transformations pour passer d'une chaîne à une autre

On considère la matrice **C**, résultat de l'algorithme de Wagner et Fisher appliqué sur les chaînes : **A='samedi'** et **B='vendredi'**

		v	e	n	d	r	e	d	i
	0	1	2	3	4	5	6	7	8
s	1	1	2	3	4	5	6	7	8
a	2	2	2	3	4	5	6	7	8
m	3	3	3	3	4	5	6	7	8
e	4	4	3	4	4	5	5	6	7
d	5	5	4	4	4	5	6	5	6
i	6	6	5	5	5	5	6	6	5

Le chemin en gras, du bas vers le haut (**5 → 5 → 5 → 5 → 4 → 3 → 2 → 1 → 0**) est l'un des chemins qui indiquent les opérations à exécuter pour transformer la chaîne **'samedi'** en la chaîne **'vendredi'**. Il suffit d'aller de case en case, en partant de la case inférieure droite de **C**, et en allant vers une case réalisant le minimum calculé dans cette case :

- un déplacement horizontal indique une insertion dans **A** ;
- un déplacement vertical indique une suppression dans **A**;
- un déplacement en diagonal indique une substitution dans **A**.

Ainsi, la séquence des transformations permettant de passer de **'samedi'** à **'vendredi'** est :

5	→	4	→	3	→	2	→	1	→	0
samedi	*remplacer*	**saredi**	*remplacer*	**sdredi**	*remplacer*	**ndredi**	*insérer*	**endredi**	*insérer*	**vendredi**

Donc, il y a **3** remplacements et **2** insertions.

Q.10- Écrire la fonction ***liste_transformations (A, B)*** qui reçoit en paramètres deux chaînes de caractères **A** et **B** non vides, et qui retourne la liste des transformations pour passer de la chaîne **A** à la chaîne **B**.

Exemple :

La fonction ***liste_transformations ('samedi', 'vendredi')*** retourne la liste :

['samedi' , 'saredi' , 'sdredi' , 'ndredi' , 'endredi' , 'vendredi']

≈≈≈≈≈≈≈≈ **FIN DE L'ÉPREUVE** ≈≈≈≈≈≈≈≈