Manipulation de texte, `Ctrl+F`

PCSI/PTSI

I Chaînes et séquences

Exercice 1. *Recherche linéaire*

Implémentez une fonction `recherche_lineaire` qui prend en entrée une chaîne caractères et un caractère et qui renvoie l'indice de la première occurrence du caractère dans la chaîne, ou -1 si le caractère n'est pas dans la chaîne.

Exercice 2. *Comptage liste*

Implémentez une fonction `compter` qui prend en paramètre une liste `L` et un objet `x` et qui renvoie le nombre de fois que `x` est dans `L`.

Exercice 3. *Comptage chaîne*

Implémentez une fonction `compter` qui prend en paramètre une chaîne `S` et un caractère `c` et qui renvoie le nombre de fois que `c` est dans `S`.

Exercice 4. *Deuxième minimum*

Implémentez une fonction `deuxieme_minimum` qui calcule le second minimum des éléments d'une liste d'entiers positifs.

Exercice 5. *Voyelles*

Implémentez une fonction `voyelles` qui prend en entrée une chaîne de caractère et renvoie une nouvelle chaîne où seules les voyelles de la chaîne originale sont conservées.

II Passage en deux dimensions

Exercice 6. *Pyramide*

Implémentez une fonction `pyramide` qui prend en entrée un entier `n` $\geqslant$ `0` et qui renvoie une pyramide d'allumettes à `n` étages en chaîne de caractères. *e.g.* pour `n = 4` :

```
   |
  |||
 |||||
|||||||
```

Exercice 7. *Table ASCII*

Implémentez une fonction `table_ascii` qui ne prend aucune entrée et renvoie la table ASCII dans une chaîne de caractère sous forme de tableau à double entrée :

	0	1	2	3	4	5	6	7	8	9
0										
10										
20										
30				!	«	#	$	%	&	’
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	‘	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{			}	~		

Exercice 8. *Table de multiplication*

Implémentez une fonction `table_multiplication` qui prend un entier en entrée et renvoie la table de multiplication de `0` à `n` dans une chaîne de caractère sous forme de tableau à double entrée. *e.g* pour `n = 4` :

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	6	8
3	0	3	6	9	12
4	0	4	8	12	16

III Rechercher encore

Il existe un grand nombre d’algorithmes ayant pour but rechercher de chaîne de caractères dans un texte. On se propose d’utiliser celui de la fenêtre glissante.

Pour chercher un mot de taille `n` dans un texte de taille `m`, on vérifie pour chaque position `i < m-n` du texte si les caractère `i`, `i+1`, …, `i+n-1` forment le mot recherché.

Exercice 9. *Rechercher à une position*

Implémentez une fonction `rechercher_i` qui prend trois entrées : une chaîne de caractère `s`, une chaîne de caractère `w` et un entier `i` et qui renvoie un booléen vrai si et seulement si le mot `w` est une sous-chaîne de `s` à l’indice `i` de `s`.

Testez votre fonction pour les entrées suivantes :

1. `s = "Nadine␣aime␣la␣grenadine", w = "dine", i = 2`
2. `s = "Nadine␣aime␣la␣grenadine", w = "dine", i = 20`
3. `s = "Nadine␣aime␣la␣grenadine", w = "dine", i = 1`
4. `s = "Nadine", w = "Nadine", i = 0`
5. `s = "Yooooohoooo␣!", w = "ooo", i = 1`
6. `s = "Yooooohoooo␣!", w = "ooo", i = 2`
7. `s = "Yooooohoooo␣!", w = "ooo", i = 8`

Exercice 10. *Rechercher à l'aide de* `recherche_i`

Implémentez une fonction `rechercher_0` qui prend trois entrées : une chaîne de caractère `s`, une chaîne de caractère `w` et un entier `d` et qui cherche le mot `w` dans le texte `s` à partir du caractère d'indice `d`. La sortie de la fonction sera l'indice le plus petit de `s` où se trouve la première lettre du mot `w` dans `s`, -1 si `w` n'est pas dans `s`.

Pour implémentez cette fonction vous utiliserez la fonction `rechercher_i` de l'exercice précédent.

Testez votre fonction pour les entrées suivantes :

1. `s = "Nadine␣aime␣la␣grenadine", w = "dine", d = 0`
2. `s = "Nadine␣aime␣la␣grenadine", w = "dine", d = 20`
3. `s = "Nadine␣aime␣la␣grenadine", w = "dine", d = 2`
4. `s = "Nadine", w = "Nadine", d = 0`
5. `s = "Yooooohoooo␣!", w = "ooo", d = 0`
6. `s = "Yooooohoooo␣!", w = "ooo", d = 2`
7. `s = "Yooooohoooo␣!", w = "ooo", d = 8`

Exercice 11. *Rechercher avec boucle imbriquée*

Implémentez une fonction `rechercher` qui prend trois entrées : une chaîne de caractère `s`, une chaîne de caractère `w` et un entier `d` et qui cherche le mot `w` dans le texte `s` à partir du caractère d'indice `d`. La sortie de la fonction sera l'indice le plus petit de `s` où se trouve la première lettre du mot `w` dans `s`, -1 si `w` n'est pas dans `s`.

Testez votre fonction pour les entrées données dans la question précédentes.

Exercice 12. *Rechercher tout*

En utilisant la fonction `rechercher` de la question précédente, implémentez une fonction `rechercher_tout` qui prend deux entrées : une chaîne de caractère `s` et une chaîne de caractère `w`. La sortie de la fonction sera la liste des indices de `s` où se trouve la première lettre d'un mot `w` dans `s`.

On fera attention que deux mots `w` *ne se superpose pas.* Si c'est le cas on prendra uniquement l'indice le plus petit. *e.g* pour le texte : `"nananadine"`, si on cherche `"nana"` on trouvera deux positions, 0 et 2. Mais la lettre en position 2 est déjà une lettre de la première occurrence de « nana », on renverra uniquement la position 0. Cela évitera les conflits lorsqu'il s'agira de remplacer le mot `"nana"`

Testez votre fonction pour les entrées suivantes :

1. `s = "Nadine␣aime␣la␣grenadine", w = "dine"`
2. `s = "Nadine", w = "Nadine"`
3. `s = "Yooooohoooo␣!", w = "ooo"`

Exercice 13. *Remplacer*

En utilisant la fonction `rechercher_tout` de la question précédente, implémentez une fonction `remplacer` qui prend trois entrées : une chaîne de caractère `s`, une chaîne de caractère `w` et une chaîne de caractère `v`. La sortie de la fonction sera le texte dans lequel toutes les occurrences de `w` seront remplacées par celle de `v`.

Il existe plusieurs manière de résoudre ce problème. Les indications ci-dessous ne sont pas forcément utiles pour toutes les solutions.

On pourra commencer par supposer que `w` *et* `v` *sont de même longueurs.* Pour ensuite gérer le cas où `v` et `w` sont de longueurs différentes on conservera le décalage induit sur les indices par les remplacements déjà effectués.

On peut passer d'une chaîne de caractère à une liste de caractère grâce à **`list`**`(s)`. On peut aussi passer d'une liste de caractère à une chaîne de caractère en faisant `''.join(L)`. On rappel qu'on peut remplacer plusieurs éléments consécutifs d'une liste par une autre liste grâce aux slices : `L[12:21] [1,2,4]` .

Testez votre code à l'aide de l'exemple suivant :

```
s8="Après␣un␣premier␣jeu ,␣les␣enfants␣jouent␣à␣un␣nouveau␣jeu"
w8="jeu"
remplacer(s8,w8,"joujou")
```

IV Fichiers

Lire la section sur les fichiers dans les éléments de cours.

Exercice 14. *Lecture d'un fichier*

1. Quel est le chemin absolu du fichier dans lequel vous travaillez ?
2. Quel est le chemin absolu du fichier `star_wars.txt` disponible dans le dossier `txt` du dossier de ressources extrait sur votre ordinateur ?
3. Quel est le chemin relatif du fichier `star_wars.txt` si on se situe dans le dossier de ressources extrait sur votre ordinateur ?

Implémentez les fonctions suivantes :

1. `nb_c` prend en entrée un chemin de fichier et renvoie le nombre de caractères écrits dans le fichier.
2. `nb_m` prend en entrée un chemin de fichier et renvoie le nombre de mots écrits dans le fichier.
3. `nb_l` prend en entrée un chemin de fichier et renvoie le nombre de lignes écrites dans le fichier.

Exercice 15. *Recherche dans un fichier*

En utilisant la console Spyder et la fonction `rechercher_tout` donnez la liste des positions où on peut trouver le mot `"nadine"` dans le fichier `"txt/texte_interressant.txt"`.

Exercice 16. *Remplacer dans un fichier*

En utilisant la console Spyder et la fonction `remplacer` remplacer toutes les occurrences du mot `"Force"` par `"Nadine"` dans le fichier `"txt/star_wars.txt"` et écrivez un fichier `"txt/nadine_wars.txt"` avec le résultat.