Devoir surveillé 1 ITC (2h) – 6 décembre 2025

- Les calculatrices sont interdites.
- Si un étudiant est amené à repérer ce qui lui semble 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.

L'objectif du sujet est d'aider le père Noël pour l'organisation de sa distribution annuelle de jouets. Les listes et dictionnaires évoqués dans les exemples se trouvent sur la dernière page.

Partie I – Gestion du stock de jouets

Depuis le début de son activité en 1855, le père Noël gère son stock de jouets avec une liste de chaînes de caractères `L` répertoriant tous les jouets de son entrepôt. Par exemple, si `L=L1` avec `L1` la liste donnée en annexe, cela signifie que l'entrepôt contient 2 camions de pompiers, 3 poupées Parpie, 2 jeux de construction Lebo... Cette représentation n'étant pas pratique, le père Noël souhaite convertir `L` en un dictionnaire `D` qui associe à chaque jouet sa quantité dans l'entrepôt. Par exemple, à partir de `L1`, il souhaite obtenir `D1`.

1. Écrire une fonction `liste_vers_dico` qui prend pour argument la liste `L` et renvoie le dictionnaire `D` correspondant.

À partir de maintenant, le stock de jouets est représenté par le dictionnaire `D` décrit ci-dessus.

1. Écrire une fonction `compterJouets` qui prend pour argument le dictionnaire `D` et renvoie le nombre de jouets dans l'entrepôt du père Noël. Par exemple, l'appel `compterJouets(D1)` renvoie 17.
2. Écrire une fonction `maxJouets` qui prend pour argument le dictionnaire `D` et renvoie le jouet le plus présent dans l'entrepôt. En cas d'égalité, on renvoie l'un des jouets les plus présents. Par exemple, pour `D1`, ce sont les poupées Parpie et les puzzles Kopémon qui sont les plus présents. L'appel `maxJouets(D1)` renvoie, par exemple, `'poupée Parpie'`.
3. Afin d'être plus précis, écrire une fonction `maxJouetsListe` qui prend pour argument le dictionnaire `D` et renvoie la liste des jouets les plus présents dans l'entrepôt. Par exemple, l'appel `maxJouetsListe` renvoie `['poupée Parpie','puzzle Kopémon']`.

Chaque jour, les lutins du père Noël construisent de nouveaux jouets, ce qui nécessite une mise à jour régulière du stock. Pour garder un historique, le père Noël fait une copie journalière du dictionnaire `D` avant sa mise à jour.

4. À l'aide d'une boucle `for`, écrire une fonction `copieD` qui prend pour argument un dictionnaire `D` qui renvoie un dictionnaire `E` contenant les mêmes clés et valeurs que `D`. *Indication : On notera qu'une « copie » de la forme* `E=D` *ne répond pas à la question car alors toute modification de* `E` *entraînerait une modification de* `D`.

À la fin de la journée, chaque lutin enregistre les jouets qu'il a construit dans un dictionnaire `d`. Par exemple, si `d={'Jeu de construction Lebo': 150, 'Livre Henry Botteur': 200}`, cela signifie que le lutin a construit 150 jeux de construction Lebo et 200 livres Henry Botteur. Les jouets construits par l'ensemble des lutins sont consignés dans une liste `listeLutins` dont chaque élément est le dictionnaire `d` associé à l'un des lutins. Étant donnés le dictionnaire `D` et la liste `listeLutins`, le père Noël souhaite mettre à jour son stock. Par exemple, à partir de `D1` et de `listeLutins1`, il obtient `D2`.

5. Écrire une fonction `ajoutProd` qui prend pour arguments le dictionnaire `D` et la liste `listeLutins` et renvoie le nouveau dictionnaire décrivant le stock. On commencera par créer une copie de `D` que l'on pourra modifier sans modifier `D`. Par exemple, `ajoutProd(D1,listeLutins1)`| renvoie `D2`.

Partie II – Réorganisation de l'entrepôt

Dans cette partie, on appelle ***caractère spécial*** un caractère ne faisant pas partie des 52 lettres minuscules et majuscules de l'alphabet (`"a"`, `"b"`, ..., `"z"`, `"A"`, `"B"`, ..., `"Z"`). Par exemple, les signes de ponctuation (`" "`, `"!"`, `"."`, ...), les chiffres (`"0"`, `"1"`, ..., `"9"`), les lettres accentuées (`"é"`, `"à"`, `"É"`, ...) sont des caractères spéciaux.

En Python, on pourra utiliser les fonctions `ord` et `chr` qui établissent une correspondance entre les chaînes de caractères de longueur 1 et les entiers et dont certaines valeurs sont données en annexe.

On note `M` la liste sans doublon contenant les noms de tous les jouets présents dans l'entrepôt du père Noël. Par exemple, si le stock est décrit par le dictionnaire `D2`, alors la liste correspondante est la liste `M1`.

6. Écrire une fonction `dico_vers_noms` qui prend pour argument un dictionnaire décrivant le stock et renvoie la liste `M` décrite ci-dessus.

Dans la suite, on dira qu'une liste `M` est ***standardisée*** si toutes ses chaînes de caractères sont non vides et ont pour premier caractère une lettre majuscule ou bien un caractère spécial. En d'autres termes, les noms commençant par l'une des 26 lettres minuscules sont interdits.

Afin de standardiser la nomenclature de son stock, le père Noël modifie la liste `M` obtenue par la fonction `dico_vers_noms` : lorsque le nom d'un jouet commence par une lettre minuscule, il la remplace par sa version majuscule. Par exemple, à partir de `M1`, il obtient `M2` en remplaçant les trois lettres minuscules par les lettres majuscules correspondantes.

7. Écrire une fonction `standardisation` qui prend pour argument une liste de chaînes de caractères et modifie chaque chaîne de caractères commençant par une lettre minuscule en remplaçant cette dernière par la lettre majuscule correspondante. Par exemple, `standardisation(M1)` renvoie `M2`. *Indication : on rappelle qu'il n'est pas*

possible de modifier directement une chaîne de caractères. On en construira donc une nouvelle. Pour concaténer deux chaînes de caractères, on utilise l'opérateur +. Par exemple, si `s` *est une chaîne de caractères,* `s[:3]+s[5:]` *est la chaîne de caractères formée des 3 premiers caractères de* `s` *suivis du sixième, septième... jusqu'au dernier.*

On suppose maintenant que la liste `M` est standardisée. Afin de gagner en productivité, le père Noël souhaite trier son stock de jouets par ordre alphanumérique (c'est-à-dire par ordre alphabétique en autorisant les caractères spéciaux). Il confie cette tâche à 27 lutins numérotés de 0 à 26. Chacun des 26 premiers lutins est associé à une lettre de l'alphabet et va gérer les jouets dont le nom commence par cette lettre. Le 27$^{\text{ème}}$ lutin va gérer les jouets dont le nom commence par un caractère spécial.

8. Écrire une fonction `repartition` qui prend pour argument une liste standardisée et renvoie une liste `R` de taille 27 telle que `R[i]` est l'ensemble des jouets attribués au lutin numéro `i` pour `i` entre 0 et 26. Par exemple, `repartition(M2)` renvoie `R1`.

Maintenant que chacun des 27 lutins est en charge d'une partie de l'entrepôt, sa mission est de trier les jouets qui lui sont attribués. Tout d'abord, il faut être capable de comparer deux chaînes de caractères suivant l'ordre alphanumérique. Pour comparer deux chaînes de caractères `s1` et `s2`, on utilise le même principe que pour l'ordre alphabétique d'un dictionnaire papier. En Python, on considérera que la caractère `c1` est avant `c2` dans l'ordre alphanumérique lorsque `ord(c1)`$\leq$`ord(c2)`.

9. Écrire une fonction `comp` qui prend pour argument deux chaînes de caractères `s1` et `s2` et renvoie

$$\begin{cases} -1 & \text{si } \texttt{s1} \text{ est strictement avant } \texttt{s2} \text{ dans l'ordre alphanumérique} \\ 0 & \text{si } \texttt{s1} \text{ et } \texttt{s2} \text{ sont égaux} \\ 1 & \text{si } \texttt{s1} \text{ est strictement après} \end{cases}$$

Par exemple

s1	"A"	"B"	""	""	"abcd"	"abc"	"abed"
s2	"b"	"a"	""	"abcd"	""	"abcd"	"abcd"
comp(s1,s2)	-1	-1	0	-1	1	-1	1

Étant donnée une liste `M` sans doublon, on souhaite construire une lite `T` ayant les mêmes éléments que `M` et triée par ordre alphanumérique. Pour cela, on va utiliser le ***tri par insertion*** dont voici le pseudo-code :

(Étape 1) Initialement, la liste `T` est vide.

(Étape 2) On parcourt tous les éléments `e` de `M`. Pour chaque `e` :

(Étape 2a) On définit une variable `i` de la manière suivante :

- si tous les éléments de `T` sont strictement avant `e` dans l'ordre alphanumérique, on pose `i = len(T)` ;
- sinon `i` est défini comme le plus petit indice tel que `comp(T[i],e)=1`.

(Étape 2b) On insère `e` à l'indice `i` dans `T`.

(Étape 3) On renvoie `T`

Voici l'évolution de ces différentes variables lors du tri de la liste `M=["Abc","Az","Abb","Abcd","Abd"]` :

	e	i	T
Initialisation			[]
Fin 1$^{\text{er}}$ tour boucle	"Abc"	0	["Abc"]
Fin 2$^{\text{e}}$ tour boucle	"Az"	1	["Abc","Az"]
Fin 3$^{\text{e}}$ tour boucle	"Abb"	0	["Abb","Abc","Az"]
Fin 4$^{\text{e}}$ tour boucle	"Abcd"	2	["Abb","Ac","Abcd","Az"]
Fin 5$^{\text{e}}$ tour boucle	"Abd"	3	["Abb","Abc","Abcd","Abd","Az"]

10. À l'aide d'une boucle `while`, écrire une fonction `indiceInsertion` qui prend pour arguments une liste `T` que l'on supposera triée et sans doublon ainsi qu'une chaîne de caractères `e` et renvoie l'entier `i` comme défini à l'étape 2a.
11. Écrire une fonction `insertion` qui prend pour arguments une liste triée et sans doublon `T`, une chaîne de caractères `e` et un entier `i` et insère l'élément `e` dans `T` à l'indice `i` (cf. étape 2b). La fonction renverra la liste obtenue après insertion et ne devra pas modifier la liste initiale `T`.
12. À l'aide des questions précédentes, écrire une fonction `triInsertion` qui prend pour argument une liste `M` sans doublon et renvoie la liste triée `T` obtenue en appliquant un tri par insertion sur `M`.

Dans le but d'améliorer légèrement le temps d'exécution de la fonction `triInsertion`, on propose de réécrire l'étape 2a à l'aide d'une recherche dichotomique.

13. Écrire une fonction `indiceInsertionDicho` qui prend pour arguments une liste `T` triée et sans doublon ainsi qu'une chaîne de caractères `e` et renvoie l'entier `i` comme défini à l'étape 2a à l'aide d'une recherche dichotomique.

Partie III - Lettres au père Noël

Soit $n \in \mathbb{N}$. Pour s'entraîner à gérer les lettres des enfants, le père Noël demande à la mère Noël de générer toutes les combinaisons possibles de jouets. Ainsi, étant donnée une liste de jouets `J` sans doublon, la mère Noël doit créer une liste de listes `Combi` dont les éléments sont des listes de taille n et représentent toutes les combinaisons possibles de n jouets. Attention, si deux listes contiennent les mêmes jouets (éventuellement dans un ordre différent), alors elles sont considérées comme identiques ; une seule de ces deux listes doit donc apparaître dans `Combi`. De plus, chaque jouet peut apparaître au plus une fois dans un éléments de `Combi`. Par exemple, si $n = 3$ et `J=J1`, alors `Combi` vaut `Combi1`.

14. Écrire une fonction `combiner` qui prend pour argument la liste `J` et l'entier `n` et renvoie la liste `Combi`. On supposera que la condition `n<=len(J)` est vérifiée.

Annexe 1 : correspondance entre entiers et caractères :

s	"A"	"B"	...	"Y"	"Z"	"a"	"b"	...	"y"	"z"
ord(s)	65	66	...	89	90	97	98	...	121	122

i	65	66	...	89	90	97	98	...	121	122
chr(i)	"A"	"B"	...	"Y"	"Z"	"a"	"b"	...	"y"	"z"

Annexe 2 : exemples sur le stock du père Noël :

```
L1 = ['Camion de pompiers','poupée Parpie',
    'Jeu de construction Lebo', 'poupée Parpie', 'poupée Parpie',
    'Manga Houane Pice', 'Écharpe bleue',
    'Jeu de société SkyDassin', 'puzzle Kopémon',
    'Jeu de société SkyDassin', 'figurine heroman',
    'Jeu vidéo Delza', 'puzzle Kopémon', 'écharpe verte',
    'Jeu de construction Lebo', 'Camion de pompiers',
    'puzzle Kopémon']
D1 = {'Camion de pompiers': 2, 'poupée Parpie': 3,
    'Jeu de construction Lebo': 2, 'Manga Houane Pice': 1,
    'Écharpe bleue': 1, 'Jeu de société SkyDassin': 2,
    'puzzle Kopémon': 3, 'figurine heroman': 1,
    'Jeu vidéo Delza': 1, 'écharpe verte': 1}

listeLutins1 = [{'Jeu de construction Lebo': 150,
        'Livre Henry Botteur': 200},
    {'Manga Houane Pice': 500},
    {'Camion de pompiers': 100,'poupée Parpie': 100,
        'Jeu de construction Lebo': 100}]
D2 = {'Camion de pompiers': 102, 'poupée Parpie': 103,
    'Jeu de construction Lebo': 252, 'Manga Houane Pice': 501,
    'Écharpe bleue': 1, 'Jeu de société SkyDassin': 2,
    'puzzle Kopémon': 3, 'figurine heroman': 1,
    'Jeu vidéo Delza': 1, 'écharpe verte': 1,
    'Livre Henry Botteur': 200}

M1 = ['Camion de pompiers', 'poupée Parpie',
    'Jeu de construction Lebo', 'Manga Houane Pice', 'Écharpe bleue',
    'Jeu de société SkyDassin', 'puzzle Kopémon', 'figurine heroman',
    'Jeu vidéo Delza', 'écharpe verte']
M2 = ['Camion de pompiers', 'Poupée Parpie',
    'Jeu de construction Lebo', 'Manga Houane Pice', 'Écharpe bleue',
    'Jeu de société SkyDassin', 'Puzzle Kopémon', 'Figurine heroman',
    'Jeu vidéo Delza', 'écharpe verte']

R1 = [[], [], ['Camion de pompiers'], [], [], ['Figurine heroman'],
    [], [], [], ['Jeu de construction Lebo',
        'Jeu de société SkyDassin', 'Jeu vidéo Delza'],
    [], [], ['Manga Houane Pice'], [], [],
    ['Poupée Parpie', 'Puzzle Kopémon'],
    [], [], [], [], [], [], [], [], [], [],
    ['Écharpe bleue', 'écharpe verte']]

J1 = ['Camion de pompiers','poupée Parpie','Jeu de construction Lebo',
    'Manga Houane Pice', 'Jeu de société SkyDassin']
Combi1 = [['Camion de pompiers', 'poupée Parpie',
        'Jeu de construction Lebo'],
    ['Camion de pompiers', 'poupée Parpie', 'Manga Houane Pice'],
    ['Camion de pompiers', 'poupée Parpie', 'Jeu de société SkyDassin'],
    ['Camion de pompiers', 'Jeu de construction Lebo',
        'Manga Houane Pice'],
    ['Camion de pompiers', 'Jeu de construction Lebo',
        'Jeu de société SkyDassin'],
    ['Camion de pompiers', 'Manga Houane Pice',
        'Jeu de société SkyDassin'],
    ['poupée Parpie', 'Jeu de construction Lebo', 'Manga Houane Pice'],
    ['poupée Parpie', 'Jeu de construction Lebo',
        'Jeu de société SkyDassin'],
    ['poupée Parpie', 'Manga Houane Pice', 'Jeu de société SkyDassin'],
    ['Jeu de construction Lebo', 'Manga Houane Pice',
        'Jeu de société SkyDassin']]
```