Name : First name : Group :

Mid-Term Exam Computer Science and Digital Society 2

SCAN

October 2025

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Duration: 30'
Documents and calculator forbidden

Warning : A program that is **badly indented**, **badly commented** or with the **wrong choice of variable names** will be **penalized** *(up to -1 point).*

General instructions : : You may use the Python functions `append()`, `len()`, `pop()`, `print()`, `in` and list concatenation (+). Other built-in functions on lists are prohibited (for example : `sum()`, `min()`, `max()`, `index()`, `insert()`, slicing, ...).

Exercise 1 Solving a simple problem

Let `mat` be any 2D **rectangular** list of **natural numbers**. E.g. :

```
1 mat = [[  1,  8, 19,  5],
2        [  9,  5,  0,  4],
3        [142,  7,  0,  4]]
```

We want a function `mat_to_index` that returns the indices `(i,j)` of the matrix in ascending order. Example of expected result for the above matrix :

```
1 l = mat_to_index(mat)
2 print(l)
3 # Display: [(1, 2), (2, 2), (0, 0), (1, 3), (2, 3), (0, 3), (1, 1), (2,
    1), (0, 1), (1, 0), (0, 2), (2, 0)]
```

And indeed, if we read the values stored in `mat` in the order of the list, then we read the values of `mat` in ascending order :

```
1 print(mat[1][2]) # Displays: 0
2 print(mat[2][2]) # Displays: 0
3 print(mat[0][0]) # Displays: 1
4 print(mat[1][3]) # Displays: 4
5 print(mat[2][3]) # Displays: 4
6 print(mat[0][3]) # Displays: 5
7 ...
```

Note : if two elements are equal, then the order of their coordinates in the list is not important *(e.g. [(1, 2), (2, 2)...] or [(2, 2), (1, 2)...] are correct).*

(Q1.1) Propose two test cases (without Python code) for `mat_to_index(mat)`, where `mat` is a rectangular 2D list of natural numbers (other than general cases). (3 points)

(Q1.2) Write the function `mat_to_index(mat)`. *The quality, efficiency, and cleanliness of your code will be taken into account.* (14 points)

(Q1.3) Calculate the worst-case complexity of your function (complexity table on the back of the exam paper). Justify your answer. (3 points)

Complexity Cheat Sheet for Python Operations

(Amortised worst case in time)

Operations	**List**	**Dict**
Add element	`l.insert(v,i)` $\mathcal{O}(n)$	`d[c] = v` $\mathcal{O}(1)$
Add first	`l.insert(v,0)` $\mathcal{O}(n)$	
Add last	`l.append(v)` $\mathcal{O}(1)$	
Remove a value	`l.remove(v)` $\mathcal{O}(n)$	
Remove first	`l.pop(0)` $\mathcal{O}(n)$	
Remove last	`l.pop()` $\mathcal{O}(1)$	
Remove a key	`l.pop(k)` $\mathcal{O}(n)$	`d.pop(k)` $\mathcal{O}(1)$
Index (Read/write)	`l[k] = 42` $\mathcal{O}(1)$	`d[k] = 42` $\mathcal{O}(1)$
Containment (value)	`if v in l` $\mathcal{O}(n)$	`if v in d.values()` $\mathcal{O}(n)$
Containment (key)		`if v in d.keys()` $\mathcal{O}(1)$

Disclaimer :
These complexities are given under the assumption of "reasonable" data: of reasonable size and without hash collisions. In the real worst case, all Dict complexties are $\mathcal{O}(n)$, but such configurations are hard to reach.

Usage reminder *(keep only bigest exponents of n)*:

$\mathcal{O}(100 \times n + n^2) = \mathcal{O}(n^2)$

$\mathcal{O}(\frac{n \times (n+1)}{2}) = \mathcal{O}(n^2)$

$\mathcal{O}(n + n \times (50 + 3 \times log(n)) \times (2 + 5 \times n)) = \mathcal{O}(n^2 log(n))$

To go further :

`https://www.geeksforgeeks.org/python/complexity-cheat-sheet-for-python-operations/`

`https://www.mathweb.fr/euclide/complexite-algorithmique/`