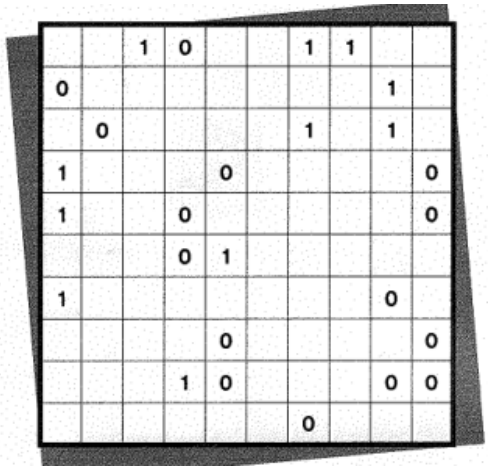
EXERCICE 5 (4 points)

Cet exercice porte sur les algorithmes et la programmation Python.



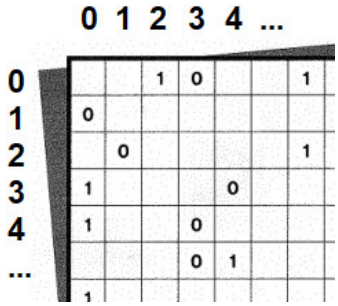
Dans le jeu du **TAKAZU**, on dispose d'une grille de 10 lignes et 10 colonnes contenant des zéros et des uns. L'objectif est de compléter les cases blanches en respectant les règles ci-dessous :

- **[REGLE1]** : chaque ligne et chaque colonne doivent contenir autant de 0 que de 1.
- **[REGLE2]** : les lignes ou colonnes identiques sont interdites.
- **[REGLE3]** : il ne doit pas y avoir plus de deux 0 ou plus de deux 1 placés à la suite, ni dans le sens vertical, ni dans le sens horizontal.

Dans cet exercice, on suppose que les valeurs de chaque ligne sont stockées dans une liste en Python, et toutes les lignes sont à leur tour placées dans une liste **globale** notée `grille`.

Ainsi, dans la situation représentée ci-contre, `grille[0][2]` vaut 1 et `grille[0][3]` vaut 0.

On décide aussi de coder par -1 toutes les cases blanches, c'est-à-dire celles dont on ne connait pas encore la valeur. Dans notre exemple, au départ `grille[0][1]` vaut -1.



1. Ecrire une fonction `autre(x)` qui renvoie 1 si `x` vaut 0 et 0 si `x` vaut 1.

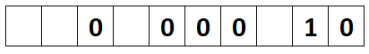
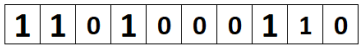
2. D'après la première règle, si une ligne contient 5 zéros, les cases blanches de cette ligne sont forcément des uns. De même si une ligne contient 5 uns, les cases blanches sont forcément des zéros.

a. Ecrire le code d'une fonction `nbValeurs(li, v)` dont les paramètres sont `li`, le numéro de la ligne de la grille et `v` la valeur que l'on souhaite compter et qui retourne la nombre de fois où la valeur `v` est présente sur la ligne `li`.

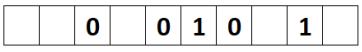
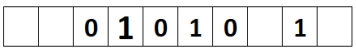
Sur l'exemple

```
>>>nbValeurs(0,1)
3
```

b. En déduire une fonction `regle1(li)` dont le paramètre `li` indique le numéro de la ligne à remplir et qui modifie l'objet `grille` en remplissant de 0 ou de 1 à partir de la première règle. Par exemple si la 6ème ligne est

 après l'appel `regle1(6)`, la ligne devient  . S'il n'y a pas assez de 0 ou de 1, la fonction ne modifie rien.

3. L'extrait de code ci-dessous effectue la recherche de deux cases identiques séparées par une blanche. Dans ce cas, on peut déterminer la valeur de la case blanche.

Par exemple, si la ligne n°4 est  , il y a deux zéros séparés par une case blanche donc la case blanche contient nécessairement un 1 sinon la troisième règle n'est pas respectée. On complètera donc en 

Recopier et compléter les ..... de la fonction `regle3(li)` dont le paramètre `li` indique le numéro de la ligne étudiée et qui modifie l'objet grille en utilisant la remarque précédente.

```
def regle3(li)
   for col in range(…):
     if grille[li][col] == grille[li][col+2] and ……:
             grille[…] = autre[…]
```

4. On suppose pour cette question que toutes les cases blanches ont été complétées. On veut déterminer si toutes les lignes sont bien différentes pour respecter la seconde règle. Chaque ligne constituée de 0 et de 1 peut être considérée comme un nombre écrit en base 2.

Ecrire le code d'une fonction `convert(L)` dont le paramètre est une liste `L` de 10 bits (0 ou 1) et renvoie la valeur décimale correspondant à la ligne `L`.

Par exemple `convert([0,0,0,0,0,1,1,1,1,1])` doit renvoyer `31`

5. On considère que l'on a stocké les valeurs obtenues à l'aide de la fonction `convert` dans une nouvelle liste `V`. On souhaite s'assurer qu'il n'y a pas de doublon.

Proposer en langage naturel l'algorithme d'une fonction dont le paramètre est une liste d'entiers et qui renvoie un booléen (VRAI ou FAUX) indiquant si cette liste possède des doublons.