université de BORDEAUX	**ANNEE UNIVERSITAIRE 2024 / 2025** **SEMESTRE 1 DS DE PRINTEMPS** **PARCOURS / ETAPE :** **Code UE : 4TPU275DSI** **Epreuve :** Programmation et Applications Interactives **Date :** 25/03/2025 **Heure :** 14h30 **Durée :** 1h30 Documents : autorisés Epreuve de M/Mme : Philippe Blasi	**Collège Sciences et technologies**

C'est un sujet à choix sur 25 points. Vous n'êtes pas obligé de tout faire pour avoir une note sur 20. Lisez tout d'abord intégralement le sujet et sélectionnez les questions qui vous semblent les plus faciles. Toutes les questions sont quasiment indépendantes et donc, n'hésitez à passer à la question suivante plutôt que de rester bloqué.

Exercice 1(8 points)

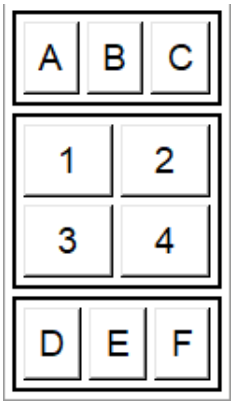
Analyser le code source ci-dessous et répondre aux questions suivantes :

A. Quel est l'affichage obtenu à l'écran lors de l'exécution du programme ?
B. Résumer (en une ou deux phrases) le rôle de chacune des fonctions **`f1`** et **`f2`** ?
C. Est-ce que la fonction **`f1`** pose problème (par exemple, une erreur à l'exécution ou alors un résultat incohérent) lorsque la valeur fournie pour **`nb_row`** est différente de celle fournie pour **`nb_col`** ? Justifier la réponse, et si cette réponse est « OUI », proposer une version modifiée de la fonction qui corrige ce problème.
D. Même question pour la fonction **`f2`**

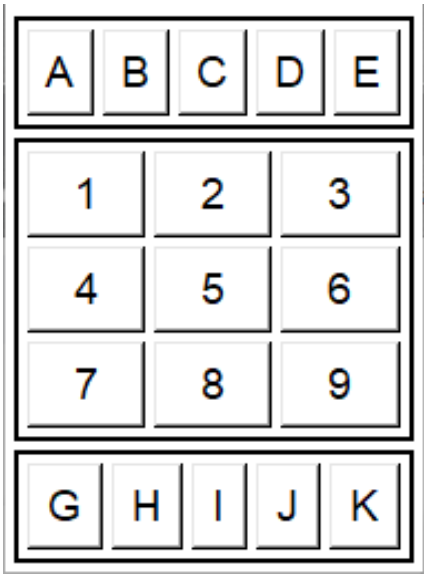
```
# ------------------------------------------------------------------------
from ezCLI import *
# ------------------------------------------------------------------------
def f1(nb_row:int, nb_col:int) -> list:
  mat = [[0 for col in range(nb_col)] for row in range(nb_row)]
  val = 0
  for row in range(nb_row):
    for col in range(nb_col):
      if row % 2 == 0:
        mat[row][col] = val
      else:
        mat[row][nb_col-col-1] = val
      val = val + 1
  return mat
# ------------------------------------------------------------------------
def f2(mat:list) -> list:
  nb_row, nb_col = len(mat), len(mat[0])
  for row in range(nb_row):
    for col in range(row):
      mat[row][col], mat[col][row] = mat[col][row], mat[row][col]
  return mat
# ------------------------------------------------------------------------
mat = f1(5,5)  # try with mat = f1(6,4)
print(grid(mat))
mat = f2(mat)
print(grid(mat))
# ------------------------------------------------------------------------
```

Exercice 2 (8 points)

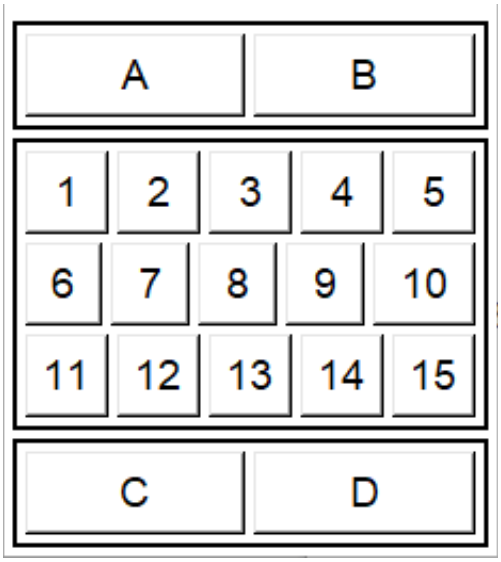
A. Écrire une fonction **`main()`** permettant d'obtenir l'interface suivante (les widgets sont tous des boutons).



B. Modifier la fonction main pour créer l'interface en fonction des paramètres suivant reçus : **`main(haut:str, bas:str, nb:int, col:int)`**. Voici deux exemples des interfaces obtenues en fonction des paramètres.



main("ABCDE","GHIJK",9,3)



main("AB","CD",15,5)

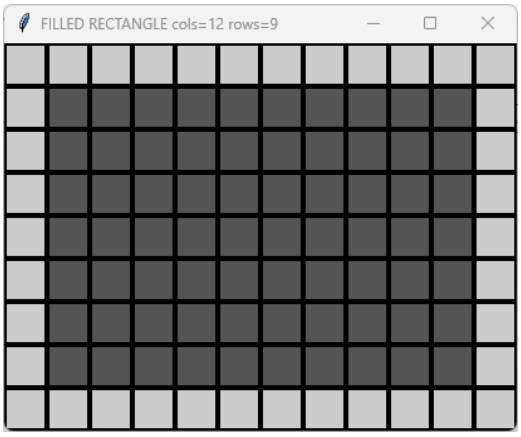
Exercice 3 (9 points)

On souhaite écrire une série de fonctions permettant d'afficher différents motifs dans des grilles rectangulaires dans une fenêtre, en utilisant la bibliothèque **`ezTK`**. La taille de la grille est définie par **`nb_row`** et **`nb_col`**, tandis que le paramètre **`size`** (de valeur 30 par défaut) définit la taille en pixels de chaque case. Les cases sont colorées soit en gris clair (couleur = **`#CCC`**) pour la couleur de fond, soit en gris foncé (couleur = **`#555`**) pour dessiner les motifs. Elles sont entourées par une bordure noire de 2 pixels de large.

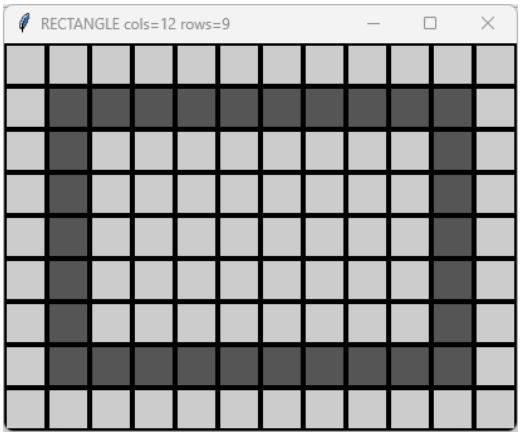
Rappels :

- Si on crée une brique avec un tuple de couleurs prévues à l'avance, le paramètre **`state`** permet de choisir la couleur à partir de son indice dans le tuple.
- On peut accéder au paramètre **`state`** d'un widget **`w`** en faisant **`w.state`**.
- On peut accéder à un widget stocké dans une grille 2D **`g`** (une fenêtre ou une frame) en utilisant ses coordonnées **`row`** et **`col`** dans la grille : **`g[row][col]`**.

A. Écrire une fonction **`filled_rectangle(nb_col:int, nb_row:int, size:int)`** permettant d'afficher un rectangle plein. Voici un exemple du résultat obtenu pour **`filled_rectangle(12,9,30)`**.



B. Écrire une fonction **`rectangle(nb_col:int, nb_row:int, size:int)`** permettant d'afficher un rectangle. Voici un exemple du résultat obtenu pour **`rectangle(12,9,30)`**.



C. Écrire une fonction **`right_rectangle(nb_col:int, nb_row:int, size:int)`** permettant d'afficher un triangle rectangle. Voici un exemple du résultat obtenu pour **`right_rectangle(12,9,30)`**.