université de BORDEAUX	**ANNEE UNIVERSITAIRE 2023 / 2024** **SEMESTRE 1 DS DE PRINTEMPS** **PARCOURS / ETAPE :** **Code UE : 4TPU275DSI** **Epreuve :** Programmation et Applications Interactives **Date :** 29/03/2024 **Heure :** 11h30 **Durée :** 1h30 Documents : autorisés Epreuve de M : Philippe Blasi	**Collège Sciences et technologies**

C'est un sujet à choix sur 28 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 (2 points chacune) :

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** ?

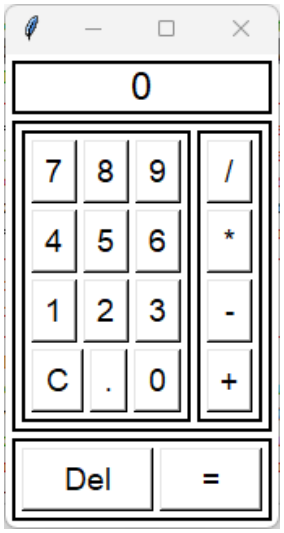
C. La fonction **f2** prend en paramètre la valeur de **n** alors que cette donnée se trouve déjà implicitement dans la variable **mat**. Proposer une version améliorée de cette fonction qui permet d 'éviter de passer **n** en paramètre.

D. Si on remplace la ligne **mat = f2(mat,7)** par la ligne **f2(mat,7)** , lequel de ces 3 résultats va-t-on obtenir ? Justifiez votre réponse.

1. Le programme ne fonctionne plus et affiche un message d'erreur.
2. Le programme fonctionne toujours mais fournit un résultat différent.
3. Le programme fonctionne toujours et fournit exactement le même résultat.

```
# ------------------------------------------------------------------------
from ezCLI import *
# ------------------------------------------------------------------------
def f1(n:int) -> list:
  return [[1 for col in range(row+1)] for row in range(n)]
# ------------------------------------------------------------------------
def f2(mat:list,n:int) -> list:
  for row in range(2,n):
    for col in range(1,row):
      mat[row][col] = mat[row-1][col-1] +  mat[row-1][col]
  return mat
# ------------------------------------------------------------------------
mat = f1(7)
print(grid(mat))
mat = f2(mat,7)
print(grid(mat))
# ------------------------------------------------------------------------
```

Exercice 2 (10 points)



On désire programmer le début d'une calculatrice.

A. (4 points) Écrire une fonction **main()** permettant d'obtenir l'interface suivante :
   1. le premier widget est un Label dans lequel les nombres entrés seront écrits. Comme il doit être facilement modifiable, il sera sauvegardé dans une variable **win.label**, si votre fenêtre s'appelle **win**. Pour accéder au contenu du label, on utilisera la propriété 'text' du Label (**win.label['text']**). Voir en bas de page les rappels sur la manipulation des chaînes de caractères.
   2. les autres widgets sont des boutons.
   3. Le bouton « C » appellera la fonction **on_C**.
   4. Le bouton « Del » appellera la fonction **on_Del**.
   5. Les boutons de chiffre et celui du « . » appelleront la fonction **on_button** avec pour paramètre le caractère du bouton.
   6. Pour les boutons restant, aucune fonction ne sera appelée.

B. (1 point) Écrire la fonction **on_C()** qui met à le label à la valeur **0**.

C. (2 points) Écrire la fonction **on_Del()** qui efface le dernier caractère ajouté au label. Attention, le label ne doit pas devenir vide. S'il devient vide, il faut le mettre à la valeur **0**.

D. (3 points) Écrire la fonction **on_button(c:str)** qui ajoute la caractère du bouton au label. Normalement la fonction 'on_button' doit simplement rajouter le caractère du bouton cliqué, derrière les caractères déjà présents dans la propriété 'text' du Label (**win.label['text']**). Mais il faudra traiter deux cas particulier avant ce cas général :

   1. Cas particulier 1 : si on appuie sur '.' alors que le Label contient déjà un caractère '.', ce deuxième '.' ne doit pas être ajouté au Label car c'est une erreur, un nombre ne peut pas avoir deux symboles de décimale. **pass** est la commande indiquant de ne rien faire.

   2. Cas particulier 2 : si on appuie sur un chiffre alors que le Label contient uniquement le caractère '0', ce chiffre remplace le '0'. Cela évite les zéros non significatifs au début du nombre entré. Si on clique successivement sur '0', '9', '6' et '2', on obtiendra dans le label '962'.

   3. Cas général : on ajoute simplement le caractère du bouton cliqué (chiffre ou '.'), derrière les caractères déjà présents dans la propriété 'text' du Label.

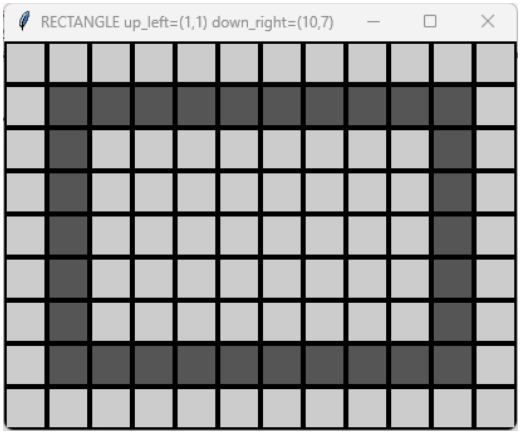
Rappels sur les chaînes de caractères :

- elles sont non mutables (modifiables). Si on désire modifier une chaîne **s**, il faut donc lui affecter une nouvelle valeur
- pour ajouter une chaîne **s2** à la fin d'une chaîne **s1**, on fait **s1 = s1 + s2** ou **s1 += s2**
- pour savoir si une chaîne **s2** est contenue dans un chaîne **s1**, on fait le test **if s2 in s1**
- pour obtenir une chaîne privée **s** de son dernier caractère, on fait **s[:-1]**
- la longueur d'une chaîne **s** est donnée par **len(s)**.

Exercice 3 (10 points)

On souhaite écrire une série de fonctions permettant d'afficher différents motifs dans des grilles rectangulaires dans une fenêtre, en utilisant au choix, la bibliothèque ezTK ou la bibliothèque tkinter. 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.

A. (4 points) Écrire une fonction
rectangle(cols:int, rows:int, size:int, ul_col:int, ul_row:int, dr_col:int, dr_row:int)
permettant d'afficher un rectangle avec pour coin supérieur gauche (up left) le point de coordonnées (**ul_col**,**ul_row**) et pour coin inférieur droit (down right) le point de coordonnées (**dr_col**,**dr_row**). Voici un exemple du résultat obtenu pour rectangle(12,9,30,1,1,10,7).



B. (6 points) Écrire une fonction
square_center(nb_col:int, nb_row:int, size:int, c_col:int, c_row:int, distance:int)
permettant d'afficher penché à 45° un carre avec un point en son centre. **c_col** et **c_row** sont les coordonnées du centre du carré et **distance** est la distance du centre aux coins. Prenez le temps de calculer les coordonnées de chacun des coins du carré à partir de celle de son centre et de la distance. Voici un exemple du résultat obtenu pour square_center(13,11,30,6,5,4).