université de BORDEAUX	**ANNEE UNIVERSITAIRE 2025 / 2026** **SESSION DE PRINTEMPS** **PARCOURS :** **Code UE : 4TPU275DSI** **Epreuve :** Programmation et Applications Interactives **Date :** 25/03/2025 **Heure :** 14h30 **Durée :** 1h30 Documents : formulaire fourni autorisé Epreuve de M : Philippe Blasi	**Collège Sciences et technologies**

C'est un sujet à choix sur 24 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:3+1+1+3)

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 ? Pas d'explication à donner, je veux uniquement ce qui s'affiche.

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 plus grand que 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.

```
# ------------------------------------------------------------------------------
from ezCLI import *
# ------------------------------------------------------------------------------
def f1(nb_row, nb_col):
    mat = [[row+col for col in range(nb_col)] for row in range(nb_row)]
    for row in range(nb_row//2):
        for col in range(nb_col):
            if col >= nb_col//2:
                mat[row][col] = mat[row][nb_col-col-1]
    return mat
# ------------------------------------------------------------------------------
def f2(mat):
    nb_row, nb_col = len(mat), len(mat[0])
    for row in range(nb_row//2):
        for col in range(nb_col):
            swap = mat[row][col]
            mat[row][col] = mat[nb_row-row-1][nb_col-col-1]
            mat[nb_row-row-1][nb_col-col-1] = swap
    return mat
# ------------------------------------------------------------------------------
mat = f1(3, 8)
print(grid(mat))
mat = f2(mat)
print(grid(mat))
# ------------------------------------------------------------------------------
```

D. Comme vu dans les exemples et les exercices de niveau A, écrire la fonction **`parser(command:str) -> str`** qui permettra de traiter la **`command`** entrée au clavier pour appeler les fonction **`f1`** et **`f2`** avec les bons paramètres et renvoyer le résultat à afficher. Une attention toute particulière devra être apportée à la gestion des entrées invalides.

Exercice 2 (8 points:2+3+3)

A. Écrire une fonction **`divide_file(name:str,name_res:str)->bool`** permettant de lire un fichier texte dont le nom est passé dans le paramètre **`name`**, de ne conserver qu'une ligne sur deux et d'écrire le résultat obtenu dans un fichier dont le nom est passé dans le paramètre **`name_res`**. Par exemple, si le fichier d'origine **`name`** contient les lignes suivante :

```
aaa
bbb
ccc
ddd
```

alors le fichier résultat **`name_res`** contiendra les lignes suivantes :

```
aaa
ccc
```

La fonction renverra un booléen indiquant si le traitement a pu avoir lieu. L'erreur générée en cas d'accès impossible à un fichier est l'erreur **`OSError`**.

B. Écrire une fonction **`suppress_comment(name:str,name_res:str,comment_symbol:str)->bool`** permettant de lire un fichier texte dont le nom est passé dans le paramètre **`name`**, de ne conserver que les lignes ne commençant pas par le symbole de commentaire **`comment_symbol`** et d'écrire le résultat obtenu dans un fichier dont le nom est passé dans le paramètre **`name_result`**. Par exemple, si le fichier d'origine **`name`** contient les lignes suivante et que **`comment_symbol`** vaut **`'#'`** :

```
aaa
# bbb
# ccc
ddd
```

alors le fichier résultat **`name_res`** contiendra les lignes suivantes :

```
aaa
ddd
```

La fonction renverra un booléen indiquant si le traitement a pu avoir lieu. L'erreur générée en cas d'accès impossible à un fichier est l'erreur **`OSError`**.

C. On désire améliorer cette fonction précédente en lui permettant de supprimer les lignes de commentaire même si le **`comment_symbol`** est précédé de caractères espace **`' '`** ou tabulation **`'\t'`**. Modifier la fonction en conséquence pour obtenir la résultat voulu. Par exemple, si le fichier d'origine **`name`** contient les lignes suivante et que

`comment_symbol` vaut `'#'` :

```
aaa
   # bbb
      # ccc
ddd
```

alors le fichier résultat `name_res` contiendra les lignes suivantes :

```
aaa
ddd
```

La fonction renverra un booléen indiquant si le traitement a pu avoir lieu. L'erreur générée en cas d'accès impossible à un fichier est l'erreur `OSError`.

Exercice 3 (8 points : 1+1+3+3)

On souhaite écrire une série de fonctions permettant d'afficher différents motifs géométriques. Pour ce faire, nous allons remplir une matrice (liste de liste) de caractères et transformer cette matrice en chaîne de caractère multi-lignes avec une fonction `grid2str(mat:list)-> str` avant de l'afficher avec la fonction `print`.

A. Écrire une fonction `get_mat(nb_row:int,nb_col:int,value) -> list` renvoyant une matrice de lignes et colonnes initialisée avec .Par exemple, `get_mat(2,3,'#')` renverra `[['#','#','#'],['#','#','#']]`.

B. Écrire une fonction `grid2str(mat:list)→ str` transformant la matrice `mat` en une chaîne de caractère multi-lignes. Par exemple, `get_mat([['A','B','C'], ['D','E','F']])` renverra `'ABC\nDEF'`., que `print` affichera sous la forme

```
ABC
DEF
```

C. Écrire une fonction `right_rectangle(nb_row:int,nb_col:int,symbol:str) -> list` renvoyant une matrice de caractère permettant d'afficher un triangle rectangle. Voici un exemple du résultat obtenu pour `print(grid2str(right_rectangle(6,12,'#'))) :`

```
#           |fin de la ligne ici
##          |fin de la ligne ici
###         |fin de la ligne ici
####        |fin de la ligne ici
#####       |fin de la ligne ici
######      |fin de la ligne ici
```

Vous ferez attention à ne pas sortir des bornes de la matrice quand le nombre de lignes et plus grand que le nombre de colonnes.

D. Écrire une fonction
`isosceles_triangle(nb_row:int,nb_col:int,symbol:str) -> list`
renvoyant une matrice de caractère permettant d'afficher un triangle isocèle. Voici un exemple du résultat obtenu pour
`print(grid2str isosceles_triangle(5,11,'#')) :`

```
     #       |fin de la ligne ici
    ###      |fin de la ligne ici
   #####     |fin de la ligne ici
  #######    |fin de la ligne ici
 #########   |fin de la ligne ici
```

Vous ferez attention à ne pas sortir des bornes de la matrice quand le nombre de colonnes est insuffisant par rapport au nombre lignes