Calculatrices et documents non autorisés, à part un aide mémoire recto-verso sur une feuille A4.

Reportez le numéro de votre groupe sur votre copie (-1 point si pas fait). Ce sujet de **3 pages** comprend des exercices indépendants. Au sein d'un même exercice, il est demandé d'utiliser des appels aux fonctions écrites dans les questions précédentes (même si vous n'avez pas donné leur code), lorsque c'est pertinent. Toute fonction doit être réalisée en Python. **Les annotations de type doivent être données**, mais **la documention n'est pas demandée** sauf si la question le précise explicitement. **Le barème est donné à titre indicatif.**

Vous ne devez pas utiliser les fonctions prédéfinies Python `count`, `split`, `join`, `del`, `map`, `sum` et d'une manière générale aucune fonction qui n'aurait pas été vue pendant les enseignements.

Vous ne devez pas utiliser de boucle while. L'utilisation de range n'est, de même, pas nécessaire.

1 Autour des colles (6 pts)

Un collégien est puni en cas de comportement répréhensible par des heures de retenues, appelées "colles". Une colle a une durée de 1 à 4 heures. Chaque colle est ainsi représentée par un entier strictement positif et inférieur ou égal à `4`.

1. On souhaite écrire une fonction `moyenne_colles` qui prend en paramètre une liste non vide d'entiers compris entre 1 et 4 inclus et qui renvoie la moyenne des valeurs de la liste. Par exemple la moyenne des valeurs de la liste `[1, 3, 2]` est `2.0`.
   a. Donner la précondition de cette fonction
   b. Donner le code de cette fonction.
2. On souhaite écrire une fonction `nb_fortes_colles` qui prend en paramètre une liste d'entiers compris entre 1 et 4 inclus et qui renvoie le nombre d'éléments supérieurs ou égaux à 3. Par exemple, `nb_fortes_colles([3, 1, 4, 2])` vaut 2.
   a. Donner des tests qui vous semblent pertinents pour cette fonction.
   b. Donner le code de cette fonction.

On souhaite écrire sur le bulletin de l'élève un message de type `str` relatif à ses colles. Étant donnée la liste de ses heures de colles, on écrira :

- rien (chaîne vide) si la liste est vide ;
- `'collé'` si la liste contient des colles, puis :
  - soit `'occasionnellement'` si la liste contient 2 ou 3 colles ;
  - soit `'fréquemment'` si la liste contient 4, 5 ou 6 colles ;
  - soit `'très souvent'` si la liste contient 7 colles ou plus ;
- puis `'pour des comportements très inadaptés'` si la moyenne des colles est supérieure ou égale à 3.

3. **On soignera particulièrement l'indentation du code**. Écrire une fonction `message` qui prend en paramètre une liste d'entiers compris entre 1 et 4 inclus et renvoie un message comme indiqué précédemment. Par exemple :

```
>>> message([])
''
>>> message([1, 2, 1, 1, 1])
'collé fréquemment'
>>> message([3, 4])
'collé occasionnellement pour des comportements très inadaptés'
```

2 Autour d'un paquet de cartes (8.5 pts)

On représente une carte à jouer par une chaîne de caractères de taille 2 qui contient :

- en première position un caractère représentant sa valeur : un chiffre parmi `'2'`, `'3'`, …, `'9'`;
- en deuxième position un caractère représentant sa couleur parmi les 4 couleurs classiques : `'D'` (*diamond*, carreau, rouge), `'H'` (*heart*, coeur, rouge), `'S'` (*spade*, trèfle, noir), `'C'` (*club*, pique, noir).

On suppose donnée la fonction `couleur` qui prend en paramètre une carte de type `str` et qui renvoie la couleur de la carte. De même on suppose donnée la fonction `valeur` qui renvoie la valeur de la carte de type entier. Par exemple :

```
>>> couleur('9S')
'S'
>>> valeur('9S')
9
```

4. Écrire un prédicat `est_rouge` qui prend en paramètre une carte de type `str` et renvoie `True` ssi cette carte est rouge. Par ex : `est_rouge('3H')` vaut `True` et `est_rouge('9S')` vaut `False`.

Dans un jeu pour enfant le but est de former des paires. Une paire de cartes est valide si :

- l'une est rouge et l'autre est noire et
- la somme de leurs valeurs vaut 11 ou la valeur absolue de la différence de leurs valeurs vaut 4 (Rappel : Python propose une fonction `abs`).

5. Écrire un prédicat `est_paire_valide` qui prend en paramètre 2 cartes `carte1` et `carte2` de type `str` et qui renvoie `True` ssi la paire contenant `carte1` et `carte2` est valide. Par exemple:

```
>>> est_paire_valide('2H', '9C')
True
>>> est_paire_valide('2H', '9D')
False
>>> est_paire_valide('3H', '7S')
True
>>> est_paire_valide('6H', '2D')
False
>>> est_paire_valide('4H', '5C')
False
```

6. Écrire une fonction qui prend en paramètre une liste de caractères représentant des valeurs de cartes et une liste de caractères représentant des couleurs de cartes et qui génère une liste de cartes énumérées comme suit :

```
>>> genere_paquet(['2', '3', '4', '5'], ['D', 'H'])
['2D', '3D', '4D', '5D', '2H', '3H', '4H', '5H']
```

7. **On soignera particulièrement l'indentation du code**. Écrire une fonction `separe_paquet` qui prend en paramètre une liste de caractères représentant la liste des couleurs des cartes d'un paquet et renvoie la liste de chaînes qui regroupe les couleurs identiques qui se suivent. Par exemple:

```
>>> separe_paquet(['D', 'D', 'S', 'C', 'C', 'C', 'D'])
['DD', 'S', 'CCC', 'D']
>>> separe_paquet([])
[]
```

On souhaite écrire un programme principal qui :

- génère un paquet de cartes de 32 cartes dont les valeurs sont comprises entre 2 et 9 incluses et les couleurs valent `'H'`, `'D'`, `'S'` ou `'C'`. L'ordre des cartes dans le paquet n'a pas d'importance et est laissé au choix ;
- demande la saisie au clavier de 2 valeurs entières comprises entre 0 et 31 (sans vérification sur les valeurs saisies, on suppose qu'elles sont correctes) ;
- affiche un message indiquant si les cartes présentes à ces indices constituent ou non une paire valide.

Par exemple :

```
Entrer un indice dans le paquet de cartes : 2
Entrer un indice dans le paquet de cartes : 2
4H et 4H n'est pas une paire valide
```

ou

```
Entrer un indice dans le paquet de cartes : 2
Entrer un indice dans le paquet de cartes : 21
4H et 7C est une paire valide
```

8. Sans ajouter de nouvelles fonctions intermédiaires, écrire une fonction principale `main` sans paramètre dont l'exécution déclenche le comportement ci-dessus.

3 Écriture de formules mathématiques (0.5 pt)

9. Traduire en Python les expressions mathématiques suivantes en utilisant les noms de variables `L`, `C` et `a`, ainsi que la variable `pi` et les fonctions `sqrt` (racine carrée) et `cos` du module `math` (l'import n'est pas demandé) :

   a. $\frac{1}{2\pi\sqrt{LC}}$

   b. $cos^4(a)$

4 Compréhension de code (5 pts)

10. Évaluer les expressions suivantes en détaillant les calculs, en précisant le nom de l'erreur éventuellement produite. En mémoire la valeur `0` est associée à la variable `x` et la valeur `4` est associée à la variable `y`.

a. `(not x < -2) and (y >= 10 or 3 < 2 or y < 7)`
b. `x > 0 and 1/x > 10`
c. `(11%4) / (15//7)`

11. Pour la suite d'instructions donnée ci-dessous et pour les valeurs en mémoire suivantes :

a. `8` pour `x`
b. `0` pour `x`

décrire l'évolution de la mémoire en indiquant le plus clairement possible les lignes des instructions exécutées et les valeurs des expressions évaluées (**ne pas faire apparaître les lignes des instructions non exécutées**). En cas d'erreur, dire quel est son type.

```
if x < 5:     #1
    TAUX = 3  #2
elif x < 10:  #3
    cte1 = 2  #4
elif x < 15:  #5
    cte1 = 1  #6
else:         #7
    cte1 = 0  #8
x = x + cte1  #9
```

12. Récrire les instructions suivantes pour qu'elles respectent les bonnes pratiques de l'UE. On suppose que la variable `incorrect` est associée à une valeur de type `bool` en mémoire, la variable `c` est associée à une valeur de type `str` et la variable `cpt` à une valeur de type `int` en mémoire.

a.
```
if incorrect:
    cpt = cpt
else:
    cpt = cpt + 1
```

b.
```
res = ''
if c.isdigit() == True:
    res = res + c
```

c.
```
def foo(n : int) -> bool:
    if n >= 0:
        return True
    else:
        return False
```

On considère la fonction suivante :

```
def mystere(chaine:str, car:str) -> str:
    """ Précondition : len(car) == 2 """
    res = ''                                #1
    k = 0                                   #2
    for elem in chaine[0:len(chaine)-1]:    #3
        res = res + elem + car[k]           #4
        k = 1 - k                           #5
    if len(chaine) > 0:                     #6
        res = res + chaine[len(chaine)-1]   #7
    return res                              #8
```

13. On considère l'appel de fonction `mystere('bio', '59')`

a. Donner la valeur de `chaine[0:len(chaine)-1]`, `chaine[len(chaine)-1]`, `car[0]` et `car[1]`.
b. Recopier et remplir le tableau de la mémoire ci-dessous de tel sorte qu'il reflète l'exécution de `mystere('bio', '59')`. **Le nombre de colonnes est donné à titre indicatif**.

res							
k							
elem							

c. Entourer clairement dans le tableau le résultat de l'appel.

14. Donner le résultat de chacun des appels suivants :

a. `mystere('s', '89')`
b. `mystere('', '89')`