NSI2 **Contrôle 01** 09/2020

Partie 1 : sur papier

Exercice 1 - Fonction d'Ackermann-Péter 4pts

Cette fonction est définie pour tout $m \in \mathbf{N}$ et tout $n \in \mathbf{N}$ par

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

Écrire la fonction A en PYTHON.

Exercice 2 - Fonction mystère 4pts

Que fait la fonction suivante ? Ne pas expliquer le code ligne par ligne, dire ce que renvoie la fonction par rapport au paramètre d'entrée.

Code Python

```
def mystery(lst : list) -> list :
    if len(lst) < 2 :
        return lst
    else :
        return [lst[-1]] + mystery(lst[1 :-1]) + [lst[0]]
```

Exercice 3 - Opérateurs bit à bit 4pts

Calculer (laisser la trace des calculs sur la copie, même si c'est un brouillon).

1. `(3 & 5)<< 4` 2pts

2. `(11 << 3)|(128 >> 2)` 2pts

Exercice 4 - Flags 4pts

On dispose d'une fonction `mouse_state` qui renvoie un `int`.
Cet `int` est en réalité toujours un entier compris entre 0 et 255 qui est une combinaison de *flags* :

Nom	Valeur	Rôle
f0	0x1	Problème de lecture
f1	0x2	Le bouton gauche est pressé
f2	0x4	Le bouton de droite est pressé
f3	0x8	Le bouton du milieu est pressé
f4	0x10	La molette est actionnée
f5	0x20	Le bouton X1 est pressé
f6	0x40	Le bouton X2 est pressé
f7	0x80	La souris est en mouvement

1. On appelle la fonction `mouse_state`, elle renvoie 134, que peut-on en déduire ? 2pts

2. En utilisant les opérateurs bit à bit, écrire une fonction `both_buttons_pressed` qui

- ne prend aucun paramètre en entrée ;

– détermine si les boutons gauche et droite de la souris sont tous les deux pressés (indépendamment des autres cas : la souris peut être en mouvement ou pas ce n'est pas un problème) et renvoie `True` si c'est le cas et `False` sinon. 2pts

Partie 2 : sur machine

Exercice 5 - Module vector_custom 4pts

Le module `vector_custom` regroupe quelques fonctions de base pour manipuler les vecteurs du plan dans un repère orthonormé. Compléter le fichier pour que l'exécution du programme de test `test_vector_custom.py` fonctionne.

On rappelle que lorsque le plan est muni d'un repère orthonormé *rep* alors étant donnés les vecteurs $\vec{v_1}\begin{pmatrix}x_1\\y_1\end{pmatrix}$, $\vec{v_2}\begin{pmatrix}x_2\\y_2\end{pmatrix}$ et le réel k on a

- $\vec{v_1} + \vec{v_2}\begin{pmatrix}x_1 + x_2\\y_1 + y_2\end{pmatrix}$
- $k\vec{v_1}\begin{pmatrix}kx_1\\ky_1\end{pmatrix}$
- $\|\vec{v_1}\| = \sqrt{x_1^2 + y_1^2}$

Exercice 6 - Remplissage récursif 4pts

Le fichier `recursive_fill.py` contient une fonction `rf` à coder. Celle ci

– prend en entrée deux `int` `x` et `y`;

– colorie récursivement les pixels noirs en pixels blancs en partant de `(x, y)`;

– ne renvoie aucune valeur de sortie.

Implémenter la fonction `rf` et tester les résultats.

Image originale.

Remplissage de l'extérieur.

Remplissage de l'intérieur.