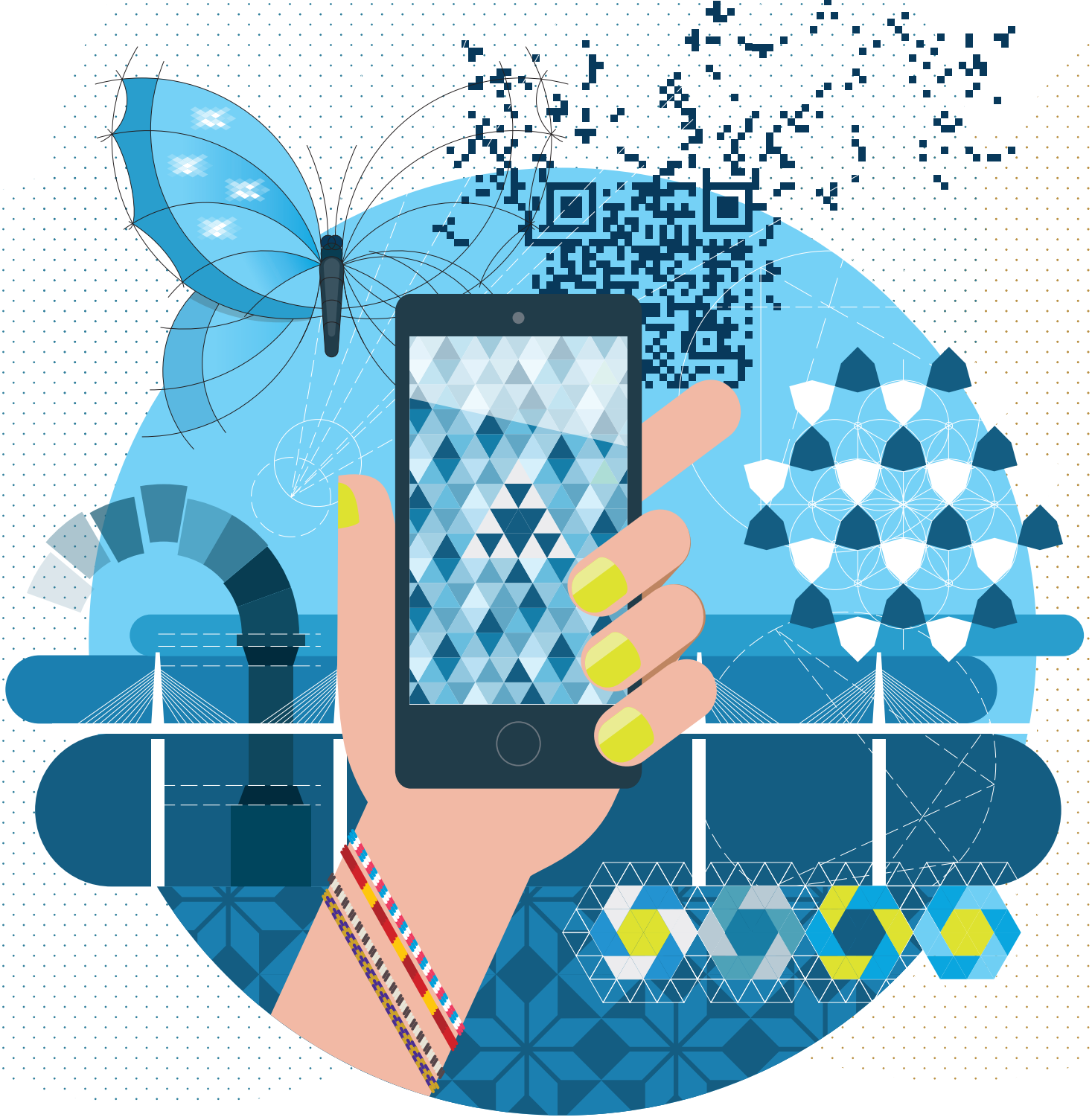
www.freemaths.fr

OLYMPIADES MATHÉMATIQUES
LYCÉE, PREMIÈRE

ACADÉMIE DE BESANÇON
2022

CORRIGÉ DE L'ÉPREUVE



22e OLYMPIADES DE MATHÉMATIQUES

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE

Sujet et Corrigé vous sont présentés par freemaths.fr . . .

CORRECTION OLYMPIADES

EXERCICES ACADÉMIQUES

BESANÇON 2022

Exercice 1 : Arithmétique « sans retenues »

Partie I : Premiers exemples et premières propriétés

1. Effectuons les additions sans retenues demandées.

En ligne pour les plus faciles : $7 \oplus 8 = 5$; $7 \otimes 8 = 6$; $12 \oplus 19 = 21$.

Pour les autres, choisissons l'option de rédiger un algorithme Python nommé « **addsr** » permettant d'effectuer des additions sans retenues.

Cet algorithme admet comme arguments les deux entiers que l'on veut additionner, écrits sous forme de listes (dont la longueur est « len(…) » en langage Python).

Le résultat est affiché sous forme de liste.

Nous laissons au lecteur le soin de reconnaître et de vérifier les résultats des additions sans retenues demandées par l'énoncé.

```
>>> def addsr(x,y):
        if len(x)>=len(y):
                u=x
                v=y
        else :
                u=y
                v=x
        for i in range(0,len(v)):
                u[len(u)-i-1]=(u[len(u)-i-1]+v[len(v)-i-1])%10
        return u

>>> addsr([8],[8])
[6]
>>> addsr([7,3,8,4,5],[9,8,5,6])
[7, 2, 6, 9, 1]
>>> addsr([9,8,5,6],[7,3,8,4,5])
[7, 2, 6, 9, 1]
>>> addsr([1,9],[1,2])
[2, 1]
>>> addsr([1,2],[1,9])
[2, 1]
>>> addsr([2,8,6,7],[4,5,6])
[2, 2, 1, 3]
>>> addsr([4,5,6],[2,8,6,7])
[2, 2, 1, 3]
```

Même option pour la multiplication sans retenue avec l'algorithme « **multsr** ».

La ligne provisoire « print(…) » permet d'afficher le détail des calculs.

Le lecteur devrait ainsi reconnaître et vérifier les résultats de certaines des multiplications sans retenues demandées par l'énoncé.

```
>>> def multsr(x,y):
        m=[0]*len(x)
        for j in range(0,len(y)):
                c=y[len(y)-j-1]
                a=[0]*len(x)
                for i in range(0,len(a)):
                        a[len(x)-i-1]=(x[len(x)-i-1]*c)%10
                a=a+[0]*j
                print("ligne numéro",j+1,a)
                m=addsr(m,a)
        return m

>>> multsr([3,8,4,5],[8,5,6])
ligne numéro 1 [8, 8, 4, 0]
ligne numéro 2 [5, 0, 0, 5, 0]
ligne numéro 3 [4, 4, 2, 0, 0, 0]
[4, 9, 0, 8, 9, 0]
*
>>> multsr([2,3],[1,9])
ligne numéro 1 [8, 7]
ligne numéro 2 [2, 3, 0]
[2, 1, 7]
>>> multsr([1,9],[2,3])
ligne numéro 1 [3, 7]
ligne numéro 2 [2, 8, 0]
[2, 1, 7]
>>> multsr([4,5,6],[2,8,6,7])
ligne numéro 1 [8, 5, 2]
ligne numéro 2 [4, 0, 6, 0]
ligne numéro 3 [2, 0, 8, 0, 0]
ligne numéro 4 [8, 0, 2, 0, 0, 0]
[8, 2, 6, 6, 1, 2]
>>> multsr([2,8,6,7],[4,5,6])
ligne numéro 1 [2, 8, 6, 2]
ligne numéro 2 [0, 0, 0, 5, 0]
ligne numéro 3 [8, 2, 4, 8, 0, 0]
[8, 2, 6, 6, 1, 2]
```

Nous avons effacé la ligne « print » de l'algorithme pour ne garder que le résultat.

Le lecteur devrait reconnaître les résultats qui manquaient.

```
>>> def multsr(x,y):
        m=[0]*len(x)
        for j in range(0,len(y)):
                c=y[len(y)-j-1]
                a=[0]*len(x)
                for i in range(0,len(a)):
                        a[len(x)-i-1]=(x[len(x)-i-1]*c)%10
                a=a+[0]*j
                m=addsr(m,a)
        return m

>>> multsr([1,2],[5,4,3])
[5, 4, 1, 6]
>>> multsr([5,4,1,6],[7])
[5, 8, 7, 2]
>>> multsr([5,4,3],[7])
[5, 8, 1]
>>> multsr([1,2],[5,8,1])
[5, 8, 7, 2]
>>> multsr([7],[8])
[6]
```

2. Le nombre 0 étant élément neutre de l'addition usuelle, aucun chiffre d'un entier *n* n'est changé lorsqu'on ajoute 0 à chacun d'eux :

Pour tout entier naturel *n* : $0 \oplus n = n \oplus 0 = n$. Nous pouvons proposer $\alpha = 0$.

3. Le nombre 1 étant élément neutre de la multiplication usuelle, aucun chiffre d'un entier *n* n'est changé lorsqu'on multiplie chacun d'entre eux par 1 :

Pour tout entier naturel *n* : $1 \otimes n = n \otimes 1 = n$. Nous pouvons proposer $\beta = 1$.

Partie II : Quelques curiosités

1. Opposés

1.a. Le chiffre des unités du résultat de chacune des additions $5+5$; $8+2$; $1+9$; $2+8$; $3+7$ est toujours 0. En conséquence : $52 \oplus 58 = 123 \oplus 987 = 0.$

1.b. Pour obtenir deux nombres à quatre chiffres opposés, nous devons choisir deux nombres dont les chiffres de même rang ont une somme égale à 10. Proposons deux exemples :

$$1234 \oplus 9876 = 0, \text{ ou bien } 7518 \oplus 3592 = 0$$

2. Carrés

2.a. $4 \otimes 4 = 6, 10 \otimes 10 = 100$ et $14 \otimes 14 = 186$

2.b. Liste des carrés pour *n* allant de 0 à 9 :

$0 \otimes 0 = 0$; $1 \otimes 1 = 1$; $2 \otimes 2 = 4$; $3 \otimes 3 = 9$; $4 \otimes 4 = 6$; $5 \otimes 5 = 5$; $6 \otimes 6 = 6$; $7 \otimes 7 = 9$;

$8 \otimes 8 = 4$; $9 \otimes 9 = 1.$

Pour *n* allant de 10 à 13, les résultats coïncident avec ceux de la multiplication usuelle qui ne met en jeu aucune retenue et il en est de même pour $n = 20$:

$10 \otimes 10 = 100$; $11 \otimes 11 = 121$; $12 \otimes 12 = 144$; $13 \otimes 13 = 169$; $20 \otimes 20 = 400.$

Liste des carrés pour *n* allant de 14 à 19 :

$14 \otimes 14 = 186$; $15 \otimes 15 = 105$; $16 \otimes 16 = 126$; $17 \otimes 17 = 149$; $18 \otimes 18 = 164$; $19 \otimes 19 = 181.$

Un algorithme Python peut nous permettre de contrôler certains des résultats.

```
>>> def listecarr():
        for i in range(10,20):
            m=multsr([1,i-10],[1,i-10])
            print("Le carré de",i,"est",m)
```

```
>>> listecarr()
Le carré de 10 est [1, 0, 0]
Le carré de 11 est [1, 2, 1]
Le carré de 12 est [1, 4, 4]
Le carré de 13 est [1, 6, 9]
Le carré de 14 est [1, 8, 6]
Le carré de 15 est [1, 0, 5]
Le carré de 16 est [1, 2, 6]
Le carré de 17 est [1, 4, 9]
Le carré de 18 est [1, 6, 4]
Le carré de 19 est [1, 8, 1]
```

3. Résolution de quelques équations.

3.a. Nous devons chercher un entier *n* à deux chiffres $n = \overline{du}$ tel que le chiffre des unités de $d + 5$ soit égal à 6 et le chiffre des unités de $u + 6$ soit égal à 5. Ce qui nous conduit nécessairement à $d = 1$; $u = 9$.

Nous proposons : $n = 19.$

3.b. Nous devons **c**hercher un entier *n* à trois chiffres $n = \overline{1du}$ tel que le chiffre des unités de $d + 7$ soit égal à 0 et le chiffre des unités de $u + 2$ soit égal à 3. Ce qui nous conduit nécessairement à $d = 3$; $u = 1$.

Nous proposons : $n = 131.$

3.c. Soit un entier *n* à deux chiffres $n = \overline{du}$. Pour être solution de l'équation $n \otimes 6 = 42$, le chiffre des unités de $u \times 6$ doit être égal à 2 et le chiffre des unités de $d \times 6$ doit être égal à 4, ce qui nous conduit nécessairement à choisir *u* égal à 2 ou à 7 et *d* égal à 4 ou à 9.

Nous proposons les quatre solutions suivantes : 42, 47, 92, 97.

Soit maintenant un entier *n* à trois chiffres $n = \overline{cdu}$. Pour être solution de l'équation $n \otimes 6 = 42$, les chiffres *d* et *u* doivent vérifier les conditions précédentes et de plus le chiffre des centaines de $c \times 6$ doit être égal à 0 ce qui nous conduit à choisir nécessairement $c = 5$.

Nous pouvons proposer n'importe lequel des nombres : 542, 547, 592 ou 597.

3.d. Il y a deux solutions à un seul chiffre : $n = 1$ et $n = 6$.

Soit un entier *n* à deux chiffres $n = \overline{du}$. Pour être solution de l'équation $n \otimes 2 = 2$, le chiffre des unités de $u \times 2$ doit nécessairement être égal à 1 ou 6 et le chiffre des unités de $d \times 2$ doit être égal à 0, donc nécessairement $d = 5$, ce qui nous conduit aux entiers : $n = 51$ et $n = 56$.

Soit un entier *n* à trois chiffres, $n = c\overline{du}$. Le chiffre *u* est nécessairement égal à 1 ou à 6, le chiffre *d* à 0 ou 5 et le chiffre *c* à 5 ce qui nous conduit aux entiers : $n = 501, 506, 551, 556$.

Nous pouvons proposer les entiers : 1, 6, 51, 56 et deux à notre choix parmi 501, 506, 551 ou 556.

NB. Nous interprétons la consigne « ayant des chiffres des unités distincts » comme signifiant « n'ayant pas tous le même chiffre des unités » car le chiffre des unités d'une solution est ou bien 1 ou bien 6.

Partie III : Diviseurs, multiples, nombres premiers

1. Diviseurs de 1

1.a. Nous obtenons : $1 \otimes 1 = 3 \otimes 7 = 9 \otimes 9 = 1$

1.b. Exploitons ces diviseurs de l'unité que sont 3, 7 et 9 en considérant les paires de « produits sans retenue » suivants : $3845 \otimes 3$ et $7 \otimes 456$, ainsi que : $3845 \otimes 7$ et $3 \otimes 456$ et que : $3845 \otimes 9$ et $9 \otimes 456$.

D'après l'associativité de la multiplication sans retenues, nous avons :

$(3845 \otimes 3) \otimes (7 \otimes 456) = 3845 \otimes (3 \otimes 7) \otimes 456 = 3845 \otimes 456 = 490890.$

Il en sera de même des deux autres paires de nombres.

Il reste à expliciter les différents produits que nous avons définis.

Effectuons les calculs avec notre algorithme Python :

```
>>> multsr([3,8,4,5],[3])
[9, 4, 2, 5]
>>> multsr([7],[8,5,6])
[6, 5, 2]
>>> multsr([9,4,2,5],[6,5,2])
[4, 9, 0, 8, 9, 0]
```

```
>>> multsr([3,8,4,5],[7])
[1, 6, 8, 5]
>>> multsr([3],[8,5,6])
[4, 5, 8]
>>> multsr([1,6,8,5],[4,5,8])
[4, 9, 0, 8, 9, 0]
```

```
>>> multsr([3,8,4,5],[9])
[7, 2, 6, 5]
>>> multsr([9],[8,5,6])
[2, 5, 4]
>>> multsr([7,2,6,5],[2,5,4])
[4, 9, 0, 8, 9, 0]
```

Nous pouvons proposer les produits : $9425 \otimes 652 = 1685 \otimes 458 = 7265 \otimes 254 = 490890.$

2. Diviseurs non triviaux de zéro.

2.a. Nous remarquons que : $2022 \otimes 5 = 0$

La réponse est oui, 2022 est un diviseur non trivial de zéro.

2.b. Soit *n* un entier tel que $25 \otimes n = 0$. Supposons que *n* soit non nul ; l'un de ses chiffres (au moins celui de plus haut rang) est donc non nul.

Soit *u* le chiffre des unités de *n*. Pour que l'égalité $25 \otimes n = 0$ soit vérifiée, il faut que les chiffres des unités des produits $2 \times u$ et $5 \times u$ soit tous les deux égaux à 0, ce qui conduit à l'unique possibilité $u = 0$.

Nous pouvons itérer le raisonnement au chiffre des dizaines de *n* et, de proche en proche, à chacun des chiffres de *n*, ce qui est contradictoire avec l'hypothèse de non-nullité de *n*.

Nécessairement, si $25 \otimes n = 0$, alors $n = 0$.

La réponse est non, l'entier 25 n'est pas un diviseur non trivial de zéro.

3. Multiples de 9.

D'après la question 1, L'entier 9 est un diviseur de l'unité, en ce sens que $9 \otimes 9 = 1$.

Pour tout entier naturel *n* nous pouvons écrire la relation : $(9 \otimes 9) \otimes n = n$.

Utilisons l'associativité de la multiplication sans retenue : $n = (9 \otimes 9) \otimes n = 9 \otimes (9 \otimes n)$.

Si nous désignons par *m* le nombre $m = 9 \otimes n$, nous obtenons la relation : $n = 9 \otimes m$.

L'entier *n* est donc un multiple de 9.

Tout entier naturel *n* est divisible par 9.

NB. Le même raisonnement montrerait que 3 et 7 ont la même propriété. En effet, pour tout entier naturel *n* nous pourrions écrire : $n = (3 \otimes 7)n = 3 \otimes (7 \otimes n) = 7 \otimes (3 \otimes n)$. Tout entier *n* est un multiple de 3 et aussi un multiple de 7.

4.a. Nous avons vu que $2 = 2 \otimes 6$. Par la même occasion : $2022 = 2022 \otimes 6$. Il existe des décompositions de 2 et de 2022 qui ne font entrer en jeu aucun des entiers 1, 3, 7 ou 9.

En ce qui concerne l'entier 17, utilisons un algorithme d'investigation systématique pour rechercher des décompositions de 17 en produits de deux nombres à deux chiffres.

```
>>> def disette():
        for i in range(1,10):
            for j in range(0,10):
                for k in range(1,10):
                    for l in range(0,10):
                        m=multsr([i,j],[k,l])
                        if m==[0,1,7]:
                            print([i,j],[k,l])

>>> disette()
[2, 9] [5, 3]
[4, 3] [5, 9]
[5, 1] [6, 7]
[5, 3] [2, 9]
[5, 7] [8, 1]
[5, 9] [4, 3]
[6, 7] [5, 1]
[8, 1] [5, 7]
```

Nous avons l'embarras du choix : $17 = 29 \otimes 53 = 43 \otimes 59 = 51 \otimes 67 = \cdots$ Il existe des décompositions de 17 qui ne font entrer en jeu aucun des entiers 1, 3, 7 ou 9.

Aucun des entiers 2, 2022 et 17 n'est un nombre premier.

4.b. Utilisons un algorithme d'investigation systématique pour détecter parmi les entiers à deux chiffres ceux qui seraient susceptibles de se prêter à une décomposition de 21 faisant intervenir 3, 7 ou 9.

Nous en obtenons trois :

$$3 \otimes 47 = 7 \otimes 63 = 9 \otimes 89 = 21.$$

Nous pouvons proposer en guise de réponse deux de ces décompositions, à notre choix.

```
>>> def decomposition():
        l={1,3,7,9}
        for x in l:
                for i in range(0,10):
                        for j in range(0,10):
                                v=multsr([x],[i,j])
                                if v==[2,1]:
                                        print(x,[i,j])
```

```
>>> decomposition()
1 [2, 1]
3 [4, 7]
9 [8, 9]
7 [6, 3]
>>> multsr([3],[4,7])
[2, 1]
>>> multsr([9],[8,9])
[2, 1]
>>> multsr([7],[6,3])
[2, 1]
```

Exercice 2 : Le jeu des différences

Partie I : Le jeu

Ecrivons un algorithme Python permettant de décrire les étapes transformant une liste $u = [a, b, c, d]$ par le jeu des différences jusqu'à son arrêt. Faisons le pari que le jeu s'arrête toujours (ce qui sera démontré en fin d'exercice) et utilisons à cet effet une boucle « while », la condition d'arrêt étant que *u* soit différent de la liste nulle (« différent de » s'écrivant « != » en langage Python).

```
>>> def jeudesdiff(a,b,c,d):
        u=[a,b,c,d]
        n=0
        while u!=[0,0,0,0]:
                n=n+1
                u=[abs(u[0]-u[1]),abs(u[1]-u[2]),abs(u[2]-u[3]),abs(u[3]-u[0])]
                print("La ligne numéro",n,"est égale à",u)
        print("la longueur N du jeu est égale à",n)
```

```
>>> jeudesdiff(7,5,3,11)
La ligne numéro 1 est égale à [2, 2, 8, 4]
La ligne numéro 2 est égale à [0, 6, 4, 2]
La ligne numéro 3 est égale à [6, 2, 2, 2]
La ligne numéro 4 est égale à [4, 0, 0, 4]
La ligne numéro 5 est égale à [4, 0, 4, 0]
La ligne numéro 6 est égale à [4, 4, 4, 4]
La ligne numéro 7 est égale à [0, 0, 0, 0]
la longueur N du jeu est égale à 7
>>> jeudesdiff(2,0,2,2)
La ligne numéro 1 est égale à [2, 2, 0, 0]
La ligne numéro 2 est égale à [0, 2, 0, 2]
La ligne numéro 3 est égale à [2, 2, 2, 2]
La ligne numéro 4 est égale à [0, 0, 0, 0]
la longueur N du jeu est égale à 4
```

1. L'exécution du programme pour $u = [7, 5, 3, 11]$ et l'affichage des résultats montre que le jeu s'arrête sur la liste nulle à la septième étape. $N(7,5,3,11) = 7$

2. . L'exécution du programme pour $u = [2, 0, 2, 2]$ et l'affichage des résultats montre que le jeu s'arrête sur la liste nulle nulle à la quatrième étape. $N(2,0,2,2) = 4$

3. *a* et *b* sont deux entiers naturels distincts.

Supposons que le quadruplet initial soit (a, b, a, b).

- Après une étape nous obtenons un quadruplet de 4 nombres égaux et non nuls, le quadruplet $(|a - b|, |a - b|, |a - b|, |a - b|)$.
- Après deux étapes nous obtenons le quadruplet $(0, 0, 0, 0)$.

En conséquence, $N(a, b, a, b) = 2.$

Supposons que le quadruplet initial soit (a, b, b, a).

- Après une étape nous obtenons le quadruplet $(|a - b|,\ 0, |a - b|, 0)$ c'est-à-dire un quadruplet du type précédent.
- Après deux étapes nous obtenons le quadruplet $(|a - b|, |a - b|, |a - b|, |a - b|)$. Le jeu s'arrêtera à la prochaine étape sur la liste nulle : $N(a, b, b, a) = 3.$

4. Les quatre nombres *a, b, c, d* sont tous des entiers naturels. Soit *M* le plus grand d'entre eux et *m* le plus petit d'entre eux, de sorte que par hypothèse, pour tout entier *x* appartenant à l'ensemble $\{a, b, c, d\}$:

$$0 \leq m \leq x \leq M$$

Pour tout couple d'entiers (x, y) appartenant à l'ensemble $\{a, b, c, d\} \times \{a, b, c, d\}$, nous pouvons écrire les inégalités : $\begin{cases} m \leq x \leq M \\ -M \leq -y \leq -m \end{cases}$. En ajoutant membre à membre ces inégalités de même sens, nous obtenons la double inégalité : $-M + m \leq x - y \leq M - m$. Nous pouvons en déduire : $|x - y| \leq M - m$.

Mais puisque *m* est un entier naturel, nous avons l'inégalité : $M - m \leq M$.

En fin de compte, pour tout couple d'entiers (x, y) appartenant à l'ensemble $\{a, b, c, d\} \times \{a, b, c, d\}$, nous obtenons l'inégalité universelle : $|x - y| \leq M$.

En particulier, il en est ainsi des quatre entiers $a' = |a - b|$; $b' = |b - c|$; $c' = |c - d|$; $d' = |d - a|$ et encore plus particulièrement du plus grand d'entre eux.

Nous avons démontré que : $\text{Max}(a', b', c', d') \leq M = \text{Max}(a, b, c, d).$

Partie II : Quelques propriétés

1.a. Les permutées de la liste (a, b, c, d) autres qu'elle-même sont les listes :

$$(b, c, d, a)\ ;\ (c, d, a, b);\ (d, a, b, c)$$

1.b. Au bout d'une étape, les listes images sont, respectivement :

- $(|a-b|, |b-c|, |c-d|, |d-a|)$ pour la liste (a, b, c, d)
- $(|b-c|, |c-d|, |d-a|, |a-b|)$ pour la liste (b, c, d, a)
- $(|c-d|, |d-a|, |a-b|, |b-c|)$ pour la liste (c, d, a, b)
- $(|d-a|, |a-b|, |b-c|, |c-d|)$ pour la liste (d, a, b, c)

Autrement dit, l'image d'une permutée est la permutée de l'image. Si, au bout d'un certain nombre d'étapes, une image est le quadruplet nul, alors il en est de même, à la même étape, des images de toutes les permutées.

Nous pouvons conclure que : $N(a, b, c, d) = N(b, c, d, a)$

2.a. Quel que soit l'entier naturel non nul k, l'image au bout de la première étape du jeu de la liste $(a + k, b + k, c + k, d + k)$ est la même que celle de la liste (a, b, c, d) puisqu'une translation de k entiers conserve les écarts. Les listes (a, b, c, d) et $(a + k, b + k, c + k, d + k)$ ont des images identiques à partir de l'issue de la première étape du jeu. Si le jeu s'arrête, les deux listes auront pour image la liste nulle à la même étape du jeu.

Il en résulte que, mis à part le cas particulier où (a, b, c, d) est déjà la liste nulle (auquel cas son numéro *N* est 0 tandis que celui de la liste translatée est 1), nous avons $N(a + k, b + k, c + k, d + k) = N(a, b, c, d)$.

2.b. Nous savons qu'une homothétie de rapport 2 multiplie les distances par 2. Les distances entre les éléments du quadruplet $(2a, 2b, 2c, 2d)$ sont doubles de celles entre leurs homologues du quadruplet (a, b, c, d).

Par le jeu des différences, l'image du quadruplet $(2a, 2b, 2c, 2d)$ est le quadruplet double de l'image de (a, b, c, d).

NB. Retenons cette propriété qui nous servira dans la résolution de la **question III.4**.

Il en résulte que par le jeu des différences, un des deux quadruplets $(2a, 2b, 2c, 2d)$ ou (a, b, c, d) a pour image le quadruplet nul à l'issue d'une certaine étape si et seulement si l'autre a pour image ce même quadruplet au même moment.

$$N(2a, 2b, 2c, 2d) = N(a, b, c, d)$$

Partie III : Le jeu s'arrête-t-il toujours ?

1. Dans cette question uniquement, on considère des triplets et non des quadruplets. Ecrivons un algorithme Python spécifique décrivant un nombre arbitraire *m* d'étapes. Nous utilisons donc une boucle « for » et non une boucle « while » (car cette fois nous ne parions pas sur un arrêt du jeu).

.**1.a.** Si $(a,b,c) = (1,2,3)$ alors au bout de cinq étapes nous obtenons l'image $(0,1,1)$ qui est le résultat déjà obtenu à l'étape numéro 2.

Le jeu va désormais être périodique, il ne s'arrête pas.

```
>>> def jeudesdiff3(a,b,c,m):
        u=[a,b,c]
        n=0
        for i in range(1,m+1):
                n=n+1
                u=[abs(u[0]-u[1]),abs(u[1]-u[2]),abs(u[2]-u[0])]
                print("La ligne numéro",n,"est égale à",u)

>>> jeudesdiff3(1,2,3,5)
La ligne numéro 1 est égale à [1, 1, 2]
La ligne numéro 2 est égale à [0, 1, 1]
La ligne numéro 3 est égale à [1, 0, 1]
La ligne numéro 4 est égale à [1, 1, 0]
La ligne numéro 5 est égale à [0, 1, 1]
```

1.b. Plus qu'une « suggestion », le contre-exemple du **1.a** démontre que, si l'on considère ce jeu des différences sur des triplets de nombres et non sur des quadruplets, il existe des cas où le jeu ne s'arrête pas.

2. Supposons que *a* soit pair et *b* impair. Il existe deux entiers naturels *a'* et *b'* tels que $\begin{cases} a = 2a' \\ b = 2b' + 1 \end{cases}$.

Alors $a - b = 2a' - (2b' + 1) = 2(a' - b') - 1$, c'est un nombre impair.

Supposons que *a* soit impair et *b* pair. Il existe deux entiers naturels *a'* et *b'* tels que $\begin{cases} a = 2a' + 1 \\ b = 2b' \end{cases}$.

Alors $a - b = (2a' + 1) - 2b' = 2(a' - b') + 1$, c'est un nombre impair.

Donc, quelles que soient les circonstances : Si *a* et *b* sont de parité différente, leur différence est impaire.

3. Modification du jeu :

3.a. Continuons le processus initié par l'énoncé jusqu'à aboutir à un « état final » de quatre nombres pairs :

Etat initial	*i*	*p*	*i*	*i*
Etape 1	*i*	*i*	*p*	*p*
Etape 2	*p*	*i*	*p*	*i*
Etape 3	*i*	*i*	*i*	*i*
Etape 4	*p*	*p*	*p*	*p*

3.b. Puisque nous avons le choix entre deux possibilités de parité pour chacun des quatre entiers, nous devons effectivement avoir $2^4 = 16$ assortiments différents. Listons les diverses possibilités :

- (p,p,p,p) ; (i,i,i,i)
- (p,p,p,i) et ses permutées (p,p,i,p) ; (p,i,p,p) ; (i,p,p,p).
- (p,p,i,i) et ses permutées (p,i,i,p) ; (i,i,p,p) ; (i,p,p,i).
- (p,i,p,i) et sa permutée (i,p,i,p)
- (p,i,i,i) et ses permutées (i,i,i,p) ; (i,i,p,i) ; (i,p,i,i) ;

3.c. Nous avons vu qu'une liste et ses permutées avaient des comportements analogues. Si l'on considère le travail effectué à la **question 3.a**, nous pouvons en déduire que :

- (p,i,i,i) et ses permutées aboutissent à (p,p,p,p) en quatre étapes.
- (p,p,i,i) et ses permutées aboutissent à (p,p,p,p) en trois étapes puisque nous avons obtenu une liste de cette catégorie à l'issue de l'étape 1.
- (p,i,p,i) et sa permutée aboutissent à (p,p,p,p) en deux étapes puisque nous avons obtenu une liste de cette catégorie à l'issue de l'étape 2.
- (i,i,i,i) aboutit à (p,p,p,p) en une seule étape.

Quel que soit le type de liste, nous aboutissons à (p,p,p,p) en quatre étapes au plus.

NB. Notre algorithme Python pouvait fournir les résultats à notre place, en représentant un nombre pair par un « 0 » et un nombre impair par un « 1 ».

```
>>> jeudiff(1,0,0,0)
La ligne numéro 1 est égale à [1, 0, 0, 1]
La ligne numéro 2 est égale à [1, 0, 1, 0]
La ligne numéro 3 est égale à [1, 1, 1, 1]
La ligne numéro 4 est égale à [0, 0, 0, 0]
La longueur du jeu est égale à 4
>>> jeudiff(1,1,0,0)
La ligne numéro 1 est égale à [0, 1, 0, 1]
La ligne numéro 2 est égale à [1, 1, 1, 1]
La ligne numéro 3 est égale à [0, 0, 0, 0]
La longueur du jeu est égale à 3
>>> jeudiff(0,1,0,1)
La ligne numéro 1 est égale à [1, 1, 1, 1]
La ligne numéro 2 est égale à [0, 0, 0, 0]
La longueur du jeu est égale à 2
>>> jeudiff(1,1,1,1)
La ligne numéro 1 est égale à [0, 0, 0, 0]
La longueur du jeu est égale à 1
```

4.a. Notons (a_0, b_0, c_0, d_0) le quadruplet initial. La question précédente montre qu'après 4 étapes au plus, nous obtenons par le jeu des différences comme image de (a_0, b_0, c_0, d_0)un quadruplet formé de quatre nombres pairs. Notons $(2a_1, 2b_1, 2c_1, 2d_1)$ ce quadruplet-image. Son image est double de celle de (a_1, b_1, c_1, d_1).

Considérons ce quadruplet (a_1, b_1, c_1, d_1). Après 4 nouvelles étapes au plus, nous obtenons par le jeu des différences un quadruplet-image de(a_1, b_1, c_1, d_1) formé de quatre nombres pairs. Notons $(2a_2, 2b_2, 2c_2, 2d_2)$ ce quadruplet. D'après le résultat de la question 2.b de la partie II, l'image de (a_0, b_0, c_0, d_0) à l'issue de la même étape est $(4a_2, 4b_2, 4c_2, 4d_2)$.

Itérons le procédé à propos du quadruplet (a_2, b_2, c_2, d_2). En au plus 4 étapes, nous trouvons un nouveau quadruplet de nombres pairs $(2a_3, 2b_3, 2c_3, 2d_3)$; l'image de (a_0, b_0, c_0, d_0) à l'issue de la même étape est $(8a_3, 8b_3, 8c_3, 8d_3)$. Nous itérerons le procédé à propos de (a_3, b_3, c_3, d_3)…

…

Au bout d'au plus 4*n* étapes nous aurons obtenu à *n* reprises des quadruplets de nombres pairs et au bout du compte une image de (a_0, b_0, c_0, d_0) de la forme $(2^n a_n, 2^n b_n, 2^n c_n, 2^n d_n)$.

4.b. Montrons que, pour *n* assez grand, il est impossible qu'un des entiers $a_n,\ b_n,\ c_n,\ d_n$ soit non nul.

Notons $M = \mathrm{Max}(a_0, b_0, c_0, d_0)$ et plus généralement notons $M_n = \mathrm{Max}(2^n a_n, 2^n b_n, 2^n c_n, 2^n d_n)$.

La **question I.4** a montré que le plus grand des nombres d'un quadruplet image était toujours inférieur ou égal au plus grand des nombres de son quadruplet antécédent.

Nous pouvons en déduire que : $M_n \leq M_{n-1} \leq \ \ldots\ \leq M$.

Or : $M_n = \mathrm{Max}(2^n a_n, 2^n b_n, 2^n c_n, 2^n d_n) = 2^n \times \mathrm{Max}(a_n,\ b_n,\ c_n,\ d_n)$.

Les entiers $a_n,\ b_n,\ c_n,\ d_n$ sont tous des entiers naturels. Si l'un au moins d'entre eux n'est pas nul, alors le nombre $\mathrm{Max}(a_n,\ b_n,\ c_n,\ d_n)$ est au moins égal à 1 et M_n est au moins égal à 2^n.

Lorsque l'entier *n* vérifie l'inégalité $2^n > M$, l'inégalité $2^n \times \mathrm{Max}(a_n,\ b_n,\ c_n,\ d_n) \leq M$ est incompatible avec l'hypothèse de non-nullité d'un au moins des entiers $a_n,\ b_n,\ c_n,\ d_n$.

Nécessairement, les entiers $a_n,\ b_n,\ c_n,\ d_n$ sont alors tous nuls.

Désignons par n_0 le premier rang pour lequel nous avons $2^{n_0} > M$. Pour $n \geq n_0$, nous avons $2^n \geq 2^{n_0} > M$ ce qui implique la nullité de tous les entiers $a_n,\ b_n,\ c_n,\ d_n$.

Nécessairement, lorsque $n \geq n_0$, les entiers $a_n,\ b_n,\ c_n,\ d_n$ sont tous nuls.

4.c. L'image du quadruplet initial est alors le quadruplet $(2^n a_n, 2^n b_n, 2^n c_n, 2^n d_n)$ qui est le quadruplet nul :

Le jeu s'arrête toujours.

NB. Nous pouvons majorer le nombre d'étapes nécessaires à l'arrêt du jeu par l'entier $4n_0$ où n_0 est le premier rang pour lequel $2^{n_0} > M$.