Défis des fous d'info.

Trouvez un entier naturel `n` tel que `n`, `n+1`, `n+2`, `n+3` aient autant de diviseurs.

Trouvez un entier naturel `n` tel que `n`, `n+1`, `n+2`, `n+3`, `n+4` aient autant de diviseurs.

Défis des pas fous d'info.

Quelques exercices pour réviser.

Voici une trentaine de listes toutes très longues.
Elles sont dans une grande liste appelée `LL`. La liste d'indice `k` est donc `LL[k]`.

> [[10, 7, 5, 4, 17, 10, 17, 9, 18, 6, 2, 2, 9, 2, 20, 9, 4, 20, 19, 17, 2, 13, 7, 4, 13, 12, 9, 20, 0, 10, 15, 13, 16, 4, 1, 13, 11, 13, 11, 12, 6, 15, 19, 7, 17, 4, 1, 8, 1, 10, 12, 17, 7, 13, 9, 10, 9, 20, 5, 6, 20, 16, 8, 7, 1, 12, 12, 10, 10, 4, 8, 8, 8, 16, 11, 10, 9, 11, 17, 15, 6, 17, 15, 13, 7, 3, 2, 13, 4, 20, 5, 5, 14, 8, 19, 10, 10, 10, 7, 9], [13, 10, 8, 10, 1, 5, 16, 13, 13, 12, 10, 14, 10, 0, 17, 16, 2, 19, 14, 3, 10, 14, 20, 6, 2, 18, 20, 19, 8, 10, 5, 9, 14, 4, 15, 14, 6, 6, 17, 7, 20, 9, 16, 14, 3, 4, 8, 1, 17, 7, 10, 17, 9, 2, 6, 4, 5, 20, 7, 20, 8, 8, 7, 16, 9, 14, 9, 11, 10, 5, 16, 8, 4, 4, 3, 2, 10, 4, 20, 20, 12, 15, 4, 5, 13, 6, 14, 2, 20, 7, 14, 12, 10, 0, 12, 0, 1, 8, 11, 10], [14, 19, 13, 7, 15, 2, 17, 6, 17, 16, 0, 0, 12, 4, 14, 6, 12, 16, 15, 2, 0, 11, 0, 17, 12, 16, 2, 15, 16, 10, 7, 13, 4, 0, 5, 5, 1, 0, 0, 16, 6, 3, 11, 0, 13, 13, 2, 15, 5, 15, 2, 6, 13, 3, 19, 18, 1, 7, 19, 14, 8, 16, 7, 20, 9, 19, 8, 15, 0, 12, 6, 20, 13, 0, 19, 0, 18, 7, 0, 6, 8, 6, 7, 8, 5, 15, 19, 3, 0, 17, 6, 13, 10, 4, 5, 8, 14, 0, 12, 13], [0, 12, 8, 20, 5, 19, 0, 3, 6, 5, 10, 8, 12, 10, 5, 15, 12, 17, 6, 4, 14, 10, 2, 7, 18, 12, 18, 14, 3, 11, 16, 18, 14, 5, 9, 12, 10, 3, 9, 12, 17, 19, 2, 7, 0, 0, 8, 4, 15, 19, 4, 10, 8, 7, 1, 16, 1, 19, 12, 5, 4, 7, 9, 18, 16, 2, 13, 3, 12, 19, 14, 11, 18, 5, 7, 3, 15, 17, 13, 14, 6, 18, 15, 3, 1, 20, 17, 14, 18, 12, 11, 17, 10, 15, 19, 15, 5, 0, 14, 7], ...
> ....[1, 11, 12, 19, 11, 10, 19, 4, 6, 10, 18, 0, 19, 16, 7, 15, 7, 3, 16, 18, 1, 18, 8, 3, 11, 9, 6, 17, 9, 16, 18, 19, 16, 8, 17, 16, 12, 3, 19, 1, 5, 13, 18, 8, 18, 5, 4, 7, 1, 0, 4, 12, 8, 18, 20, 7, 5, 17, 2, 0, 6, 10, 13, 6, 13, 0, 18, 10, 8, 15, 5, 7, 16, 10, 3, 14, 4, 7, 5, 17, 16, 19, 20, 14, 16, 6, 15, 17, 3, 15, 18, 11, 4, 13, 2, 12, 9, 18, 20, 6]]
>
> Elles sont complètes en dernière page.

Ne les recopiez pas bêtement, allez les chercher sur le PDF de Cahier de Prépas, profitez de la fonction « copie/coller » pour les importer dans `IDLE`. [1]
Vérifiez en demandant alors la valeur de `len(LL)`.
Que donne en revanche `[len(L) for L in LL]` ?

Comparez avec

```
for k in range(len(L)) :
....print(len(L[k])
```

que vous auriez sans doutes tapé plus spontanément.
Et que pensez vous de `[len(LL[k]) for k in range(len(LL))]`.
Et de `[[k, len(LL[k])] for k in range(len(LL))]` (avec deux niveaux de crochets).

A laquelle des listes manque un élément ? (on va dire que c'est à L[8] qu'il en manque un, mais c'est juste pour mes explications).
Ajoutez lui un `10` en première position.

Pour ajouter un `10` en première position d'une liste `L`, que fait on ?

```
L.append(10)
L.insert(10)
L.inster(10,0)
L = [10]+L
L[0] = 10
```

1. sinon, importez `randrange` du module `random` et créez `[[randrange(21) for k in range(100)] for n in range(30)]`

Laquelle de ces listes a le plus grand total ?

Laquelle de ces listes a la plus grande moyenne ?

```
for k in range(len(LL)) :
....for i in range(len(L[k]) :
........s += L[k][i]
....if s > m :
........index = k
........m = s
```

```
s = 0
for k in range(len(LL)) :
....for i in range(len(L[k]) :
........s += L[k][i]
....if s > m :
........index = k
........s = m
```

```
for k in range(len(LL)) :
....s = 0
....for i in range(len(L[k]) :
........s = L[k][i]
....if s > m :
........index = k
........m = s
```

```
m, s = 0, 0
for k in range(len(LL)) :
....for i in range(len(L[k]) :
........s += L[k][i]
....if s > m :
........index = k
........s = m
....s = 0
```

```
m, index = 0, 0
for k in range(LL) :
....for i in range(len(LL) :
........s += L[k][i]
........if s > m :
............index = k
............m = s
```

Laquelle de ces listes contient le plus de `20` ?
Si vous voulez tricher, il existe la méthode `count`. Sinon, vous la reconstruisez.

De même, à la question précédente, on pouvait tricher avec la fonction `sum`.
Distinction : `sum`, `len`, `max`, `min`, `sorted` sont des fonctions (syntaxe `sum(L)`, `sorted(L)`).
`append`, `count`, `insert`, `sort` sont des méthodes (syntaxe `L.sort( )`, `L.insert(5, 2)`) et peuvent utiliser des paramètres (`L.append(10)`).

Si on édite `[L.count(20) for L in LL]`, on voit qu'une même valeur peut être atteinte plusieurs fois.
Expliquez le programme laid suivant :

```
R = [ ]
m = max([L.count(20) for L in LL])
for k in range(len(LL)) :
....if LL[k].count(20) == m :
........R.append(k)
print(R)
```

On appelle montée dans une liste `L` une étape `k` vérifiant `L[k+1] > L[k]`.
Quelle liste a le plus de montées ?
Que pensez vous de

```
def montees(L) :
....for k in range(len(L)) :
........if L[k+1] > L[k] :
............compteur += 1
....return compteur
```

```
def montees(L) :
....for k in range(len(L)) :
........if L[k] > L[k-1] :
............compteur += 1
....return compteur
```

```
def montees(L) :
....compteur = 0
....for k in range(len(L)) :
........if L[k+1] > L[k] :
............compteur += 1
........return compteur
```

Que mesure ce qui suit, et quelle liste l'emporte ?

```
def vera(L) :
....m = 0
....for k n range(len(L)-1) :
........if L[k+1]-L[k] > m :
............m = L[k+1}- L[k]
....return(k)
```

Avez vous quelque-chose à changer pour que ce programme soit utile ?

L'objectif est maintenant de trouver la plus longue phase montante sur une liste.
Voici un exemple sur une liste numérique simple :

4	3	2	5	8	9	8	9	10	12	13	15	8	9	13	11	9	10	12	7
		longueur 3				longueur 5						longueur 3				longueur 3			

Vous allez trouver la plus grande de ces longueurs en parcourant une seule fois la liste et en comptant sur vos doigts.
Tant que ça monte, vous comptez une unité de plus sur vos doigts.
Si ça retombe, vous remettez le compteur de vos doigts à zéro.

Inspirez vous de

```
for k in range(len(L)-1) :
....if L[k+1] > L[k] :
........c += 1
....else :
........c = 0
```

Mais comment retenir la plus longue montée ?

Maintenant, quelle est la liste dont les sommes partielles (c'est $A_n = \sum_{k=0}^{n} a_k$) sont les plus rapides à atteindre 100 ?
Ici, on sommer au fur et à mesure, et s'arrêter dès qu'on dépasse 100

suite	1	5	4	8	10	6	12	8	14	16	8	14	4	12	8
somme partielle	1	6	10	18	28	34	46	54	68	84	92	106	$\leftarrow$ stop		

Il faut donc créer un accumulateur et un index `k` qui avance. Mais on ne va pas le faire avancer avec une boucle `for`, car on ne sait justement pas à l'avance combien de termes on va sommer.
L'instruction sera donc une boucle conditionnelle (`while`).

```
def depasser(L) :
....partielle = 0
....while partielle < 100 :
........partielle += L[k]
....return partielle
```

```
def depasser(L) :
....k, partielle = 0, 0
....while partielle < 100 :
........partielle += L[k]
....return k
```

```
def depasser(L) :
....partielle
....while partielle < 100 :
........partielle += L[k]
....return partielle
```

Mais pourquoi se limiter à la seule valeur `100` ? Il faut pouvoir l'appliquer pour `400`, `240` ou toute autre valeur.
On va donc définir `def dépasse(L, valeur) :`

Laquelle des listes met le plus de temps à dépasser la valeur `400` ?

Remarque en passant.
On sait que quand même, le plus souvent on l'utilisera pour la valeur `100`.
Alors on peut utiliser une valeur par défaut `def depasse(L, valeur = 100) :` (programme à définir).
Vous pouvez alors solliciter `depasse(LL[0], 400)`, `depasse(LL[0], 400)` et il faut savoir que si vous tapez juste `depasse(LL[0])`, ce sera alors `depasse(LL[0], 100)` qui sera compris par Python.

Mais si la somme partielle n'atteint jamais la valeur demandée, que se passe-t-il ?
Testez `depasse(LL[18], 10000)`.

Déjà, que répondre si la valeur n'est jamais atteinte même en additionnant tous les termes de la liste ? Rien ? La longueur de a liste ? Le nombre 0 ? Autre chose ?
Il existe une solution avec boucle `for` et une solution avec boucle `while`.

```
def depasse(L, valeur) :
....somme = 0
....for k in range(len(L)) :
........somme += ....
........if somme >= valeur :
............return ....
....return ....
```

```
def depasse(L, valeur) :
....somme, 0 = 0, 0
....while k < len(L) and somme < valeur :
........somme += ....
........k += ....
....if ....
........return ....
....else :
........return .....
```

Si vous ne saisissez pas bien, exécutez les après avoir ajouté une ligne `print(somme, k)` au bon endroit.

Remplacer tous les `1` de toutes les listes par des `0` et indiquer combien il a fallu en changer.

Appliquez l'algorithme suivant à la liste `LL[0][:20]` (mais au fait, c'est quoi ça ?).
Expliquez.

```
def bulle(L) :
....reste = True
....while reste :
........reste = False
........for k in range(len(L)-1) :
............if L[k] > L[k+1} :
................L[k], L[k+1] = L[k+1], L[k]
................print(L)
................reste = True
....return L
```

Et si vous remplacez `if L[k] > L[k+1} :` par `if L[k] < L[k+1} :`.

Et si vous remplacez `if L[k] > L[k+1} :` par `if L[k] >= L[k+1} :`.

Et si vous remplacez `L[k], L[k+1] = L[k+1], L[k]` par `L[k] = L[k+1]`
`L[k+1], L[k]` ?

Défi des très fous d'info

Trouver un motif présent dans les deux premières listes (vérifier pour `[10, 17, 9]`).
Trouver le plus long motif présent dans les deux premières listes.

[[10, 7, 5, 4, 17, 10, 17, 9, 18, 6, 2, 2, 9, 2, 20, 9, 4, 20, 19, 17, 2, 13, 7, 4, 13, 12, 9, 20, 0, 10, 15, 13, 16, 4, 1, 13, 11, 13, 11, 12, 6, 15, 19, 7, 17, 4, 1, 8, 1,
10, 12, 17, 7, 13, 9, 10, 9, 20, 5, 6, 20, 16, 8, 7, 1, 12, 12, 10, 10, 4, 8, 8, 8, 16, 11, 10, 9, 11, 17, 15, 6, 17, 15, 13, 7, 3, 2, 13, 4, 20, 5, 5, 14, 8, 19, 10, 10, 10,
7, 9], [13, 10, 8, 10, 1, 5, 16, 13, 13, 12, 10, 14, 10, 0, 17, 16, 2, 19, 14, 3, 10, 14, 20, 6, 2, 18, 20, 19, 8, 10, 5, 9, 14, 4, 15, 14, 6, 6, 17, 7, 20, 9, 16, 14, 3, 4,
8, 1, 17, 7, 10, 17, 9, 2, 6, 4, 5, 20, 7, 20, 8, 8, 7, 16, 9, 14, 9, 11, 10, 5, 16, 8, 4, 4, 3, 2, 10, 4, 20, 20, 12, 15, 4, 5, 13, 6, 14, 2, 20, 7, 14, 12, 10, 0, 12, 0, 1, 8,
11, 10], [14, 19, 13, 7, 15, 2, 17, 6, 17, 16, 0, 0, 12, 4, 14, 6, 12, 16, 15, 2, 0, 11, 0, 17, 12, 16, 2, 15, 16, 10, 7, 13, 4, 0, 5, 5, 1, 0, 0, 16, 6, 3, 11, 0, 13, 13, 2,
15, 5, 15, 2, 6, 13, 3, 19, 18, 1, 7, 19, 14, 8, 16, 7, 20, 9, 19, 8, 15, 0, 12, 6, 20, 13, 0, 19, 0, 18, 7, 0, 6, 8, 6, 7, 8, 5, 15, 19, 3, 0, 17, 6, 13, 10, 4, 5, 8, 14, 0,
12, 13], [0, 12, 8, 20, 5, 19, 0, 3, 6, 5, 10, 8, 12, 10, 5, 15, 12, 17, 6, 4, 14, 10, 2, 7, 18, 12, 18, 14, 3, 11, 16, 18, 14, 5, 9, 12, 10, 3, 9, 12, 17, 19, 2, 7, 0, 0, 8,
4, 15, 19, 4, 10, 8, 7, 1, 16, 1, 19, 12, 5, 4, 7, 9, 18, 16, 2, 13, 3, 12, 19, 14, 11, 18, 5, 7, 3, 15, 17, 13, 14, 6, 18, 15, 3, 1, 20, 17, 14, 18, 12, 11, 17, 10, 15, 19,
15, 5, 0, 14, 7], [9, 2, 16, 9, 9, 7, 16, 16, 5, 5, 5, 8, 3, 3, 3, 20, 18, 10, 18, 3, 2, 13, 20, 19, 0, 10, 11, 13, 9, 0, 14, 19, 2, 3, 2, 20, 11, 17, 6, 13, 8, 5, 7, 19, 11,
11, 13, 12, 17, 5, 13, 3, 8, 0, 3, 9, 12, 11, 4, 17, 16, 5, 16, 14, 15, 12, 13, 0, 7, 6, 0, 3, 17, 14, 12, 5, 9, 13, 4, 6, 3, 1, 6, 16, 20, 9, 14, 11, 15, 19, 2, 3, 4, 11, 0,
7, 5, 1, 20, 8], [8, 17, 14, 7, 10, 5, 1, 12, 5, 11, 10, 20, 0, 20, 1, 19, 9, 18, 10, 5, 4, 3, 6, 15, 5, 6, 7, 9, 17, 16, 13, 10, 18, 0, 6, 14, 6, 13, 17, 20, 11, 8, 0, 13, 17,
5, 0, 19, 19, 6, 2, 20, 19, 8, 4, 7, 2, 8, 14, 16, 5, 20, 2, 14, 19, 16, 12, 16, 14, 4, 18, 12, 20, 2, 11, 0, 5, 8, 8, 19, 5, 12, 2, 8, 7, 2, 4, 17, 6, 16, 14, 11, 12, 16, 2,
18, 16, 8, 13, 6], [14, 9, 9, 19, 8, 2, 19, 2, 14, 20, 10, 20, 15, 20, 0, 2, 7, 17, 7, 7, 11, 14, 6, 19, 10, 20, 12, 15, 2, 3, 6, 20, 13, 10, 15, 3, 8, 5, 8, 11, 10, 1, 19,
14, 4, 10, 18, 18, 8, 7, 2, 0, 12, 9, 5, 14, 5, 3, 4, 14, 8, 18, 12, 19, 4, 7, 4, 14, 3, 1, 14, 20, 1, 20, 0, 11, 0, 9, 11, 11, 2, 19, 12, 12, 6, 15, 12, 10, 8, 11, 13, 4, 18,
17, 8, 0, 19, 8, 10], [14, 2, 3, 15, 18, 15, 8, 19, 17, 16, 1, 0, 17, 16, 3, 17, 10, 10, 11, 6, 13, 13, 12, 11, 3, 8, 18, 7, 11, 11, 10, 10, 3, 3, 11, 8, 4, 0, 17, 18, 13,
14, 9, 4, 13, 9, 7, 1, 0, 10, 1, 17, 18, 5, 7, 7, 8, 9, 18, 0, 0, 11, 5, 7, 19, 9, 11, 19, 10, 2, 0, 13, 1, 11, 7, 1, 17, 20, 7, 1, 8, 6, 17, 15, 8, 15, 20, 13, 15, 15, 14, 13,
0, 14, 8, 4, 14, 14, 7, 10], [7, 12, 3, 20, 10, 16, 13, 19, 18, 13, 1, 12, 13, 4, 6, 6, 16, 14, 15, 13, 19, 12, 15, 8, 11, 18, 20, 8, 1, 18, 3, 8, 2, 7, 4, 17, 10, 17, 16, 3,
20, 3, 9, 19, 1, 13, 0, 16, 19, 17, 17, 4, 1, 14, 13, 2, 7, 11, 12, 11, 12, 16, 16, 16, 5, 6, 8, 1, 12, 17, 8, 18, 2, 0, 17, 13, 11, 13, 20, 13, 11, 20, 19, 7, 6, 8, 7, 17,
2, 20, 15, 15, 12, 11, 2, 11, 9, 20, 1, 4], [8, 5, 5, 19, 14, 12, 18, 2, 9, 12, 6, 13, 11, 18, 3, 7, 17, 6, 0, 1, 13, 10, 8, 5, 3, 2, 14, 2, 6, 16, 11, 15, 4, 12, 2, 6, 14, 5,
14, 15, 11, 16, 14, 4, 0, 18, 0, 14, 12, 11, 11, 19, 7, 5, 9, 9, 13, 4, 13, 5, 10, 11, 14, 3, 17, 9, 10, 16, 15, 12, 10, 3, 7, 12, 10, 4, 8, 17, 13, 13, 19, 19, 14, 17, 2,
17, 6, 19, 11, 14, 2, 16, 18, 9, 14, 0, 3, 14, 3, 5], [3, 17, 4, 3, 9, 5, 5, 3, 2, 18, 7, 5, 19, 7, 14, 3, 19, 5, 8, 1, 14, 18, 5, 19, 6, 5, 14, 11, 17, 15, 5, 9, 7, 0, 1, 6, 13,
3, 4, 6, 8, 6, 5, 20, 8, 17, 7, 7, 19, 7, 1, 18, 4, 19, 0, 9, 3, 17, 16, 20, 0, 2, 7, 16, 11, 16, 11, 17, 9, 1, 9, 0, 7, 8, 20, 19, 6, 3, 12, 9, 3, 1, 17, 14, 14, 19, 8, 11, 14,
19, 19, 16, 17, 5, 9, 17, 2, 7, 20, 20], [15, 11, 0, 0, 18, 0, 0, 1, 3, 19, 19, 7, 8, 6, 12, 7, 6, 11, 17, 13, 9, 15, 19, 4, 1, 13, 11, 19, 12, 4, 10, 3, 11, 15, 20, 15, 8, 8,
16, 15, 6, 5, 12, 17, 0, 6, 11, 5, 17, 1, 14, 11, 5, 6, 0, 20, 1, 10, 13, 4, 12, 12, 0, 13, 8, 7, 5, 6, 6, 18, 3, 3, 9, 8, 0, 16, 14, 18, 16, 3, 19, 0, 5, 7, 0, 2, 9, 8, 17, 4,
1, 17, 14, 3, 1, 16, 13, 14, 16, 4], [15, 13, 4, 8, 16, 19, 3, 13, 19, 9, 2, 4, 18, 12, 11, 17, 2, 16, 18, 3, 16, 7, 9, 16, 8, 18, 10, 14, 9, 8, 16, 5, 0, 19, 8, 15, 3, 13,
19, 4, 13, 2, 17, 3, 18, 2, 1, 18, 16, 8, 4, 5, 17, 9, 6, 0, 0, 19, 17, 17, 19, 6, 16, 20, 15, 9, 20, 1, 16, 2, 11, 7, 0, 20, 15, 1, 2, 17, 18, 13, 14, 11, 12, 11, 20, 0, 10,
17, 15, 10, 1, 11, 17, 3, 20, 7, 19, 7, 9, 5], [18, 6, 6, 4, 11, 20, 6, 16, 8, 16, 4, 16, 20, 10, 17, 2, 9, 18, 0, 5, 0, 4, 15, 16, 14, 2, 9, 18, 12, 7, 17, 5, 18, 10, 20, 9,
14, 5, 4, 5, 12, 14, 17, 14, 5, 0, 13, 13, 2, 16, 7, 15, 2, 9, 0, 7, 20, 4, 6, 9, 0, 2, 8, 19, 9, 13, 7, 16, 7, 19, 16, 13, 15, 6, 16, 13, 6, 10, 0, 14, 7, 9, 1, 1, 18, 20, 17,
9, 8, 11, 18, 8, 9, 3, 6, 11, 9, 19, 7, 8], [7, 13, 2, 0, 5, 6, 17, 9, 2, 6, 0, 0, 12, 12, 3, 18, 4, 14, 13, 2, 12, 14, 6, 6, 16, 14, 20, 11, 16, 18, 13, 13, 16, 8, 10, 13, 12,
13, 15, 5, 13, 4, 2, 6, 10, 1, 4, 19, 4, 5, 14, 10, 18, 8, 0, 0, 0, 2, 17, 12, 19, 11, 16, 13, 14, 0, 9, 14, 11, 1, 9, 5, 6, 1, 20, 5, 1, 16, 14, 9, 4, 8, 13, 16, 16, 13, 14,
4, 20, 0, 1, 9, 4, 4, 0, 14, 19, 17, 9, 17], [3, 6, 17, 8, 0, 11, 13, 14, 16, 16, 6, 10, 7, 2, 8, 1, 6, 0, 8, 3, 15, 18, 15, 4, 15, 15, 7, 15, 11, 10, 1, 9, 20, 16, 7, 13, 12,
7, 11, 20, 14, 18, 16, 19, 12, 19, 6, 18, 17, 13, 15, 7, 7, 14, 0, 14, 16, 9, 11, 7, 19, 4, 17, 0, 2, 3, 4, 16, 9, 16, 0, 0, 5, 11, 2, 0, 11, 12, 16, 7, 15, 14, 17, 13, 18,
10, 3, 2, 10, 6, 10, 14, 19, 0, 14, 20, 11, 5, 20, 5], [20, 7, 15, 10, 7, 12, 13, 8, 2, 16, 16, 11, 10, 10, 5, 9, 13, 0, 13, 5, 2, 14, 0, 15, 19, 14, 17, 18, 20, 20, 8, 17,
11, 7, 18, 0, 6, 14, 13, 9, 17, 20, 6, 9, 9, 5, 14, 16, 8, 0, 19, 17, 4, 18, 1, 17, 19, 8, 0, 3, 17, 0, 2, 0, 4, 7, 6, 0, 4, 0, 18, 11, 9, 10, 11, 1, 7, 12, 20, 10, 13, 19, 20,
16, 18, 6, 18, 5, 11, 0, 1, 10, 0, 5, 10, 7, 10, 20, 17, 9], [3, 20, 14, 6, 0, 2, 14, 18, 1, 9, 2, 10, 17, 3, 13, 4, 19, 9, 16, 14, 15, 17, 8, 11, 13, 12, 7, 19, 18, 7, 16, 6,
8, 12, 15, 7, 15, 16, 2, 5, 6, 6, 3, 17, 7, 6, 20, 20, 9, 6, 9, 18, 0, 5, 12, 14, 16, 5, 18, 12, 0, 7, 20, 16, 17, 11, 19, 7, 4, 1, 3, 4, 19, 20, 2, 1, 14, 20, 5, 9, 0, 0, 14,
13, 18, 12, 0, 0, 0, 7, 2, 10, 15, 9, 18, 2, 13, 7, 6, 9], [4, 17, 6, 13, 8, 12, 12, 10, 18, 2, 11, 4, 13, 4, 5, 1, 3, 4, 0, 15, 12, 7, 20, 3, 10, 5, 7, 4, 9, 8, 12, 9, 15, 15,
12, 13, 15, 13, 17, 4, 8, 12, 14, 17, 5, 15, 7, 10, 9, 11, 8, 6, 14, 14, 14, 2, 10, 3, 4, 4, 0, 7, 3, 15, 9, 1, 10, 6, 1, 15, 13, 10, 7, 4, 13, 1, 14, 5, 19, 8, 18, 5, 0, 6,
13, 18, 11, 1, 7, 10, 9, 7, 7, 14, 4, 5, 14, 8, 5, 1], [16, 6, 8, 15, 12, 9, 2, 14, 17, 11, 16, 3, 17, 14, 5, 13, 9, 9, 10, 17, 5, 1, 11, 3, 15, 14, 13, 4, 15, 2, 20, 13, 16,
9, 15, 7, 14, 11, 9, 12, 3, 19, 8, 14, 15, 12, 10, 7, 12, 12, 3, 20, 20, 4, 5, 16, 4, 20, 16, 1, 14, 18, 19, 1, 15, 20, 4, 16, 15, 10, 1, 12, 10, 16, 9, 15, 12, 14, 7, 3,
18, 7, 8, 11, 10, 17, 12, 17, 17, 14, 10, 1, 11, 0, 8, 16, 20, 13, 13, 0], [19, 5, 16, 14, 19, 5, 4, 3, 16, 10, 10, 16, 9, 4, 14, 17, 6, 12, 19, 7, 15, 12, 12, 12, 8, 5, 18,
8, 6, 19, 7, 6, 0, 16, 19, 3, 15, 15, 16, 6, 16, 9, 7, 11, 18, 0, 13, 2, 5, 5, 5, 4, 6, 4, 20, 4, 19, 14, 3, 16, 6, 1, 16, 3, 20, 6, 11, 18, 9, 16, 5, 6, 11, 9, 15, 2, 7, 13,
14, 10, 18, 10, 10, 19, 17, 14, 14, 19, 17, 7, 10, 1, 20, 5, 7, 4, 17, 8, 10, 13], [14, 9, 19, 20, 12, 4, 4, 15, 8, 20, 1, 13, 15, 6, 13, 2, 15, 4, 19, 12, 4, 13, 15, 1, 11,
3, 4, 10, 20, 5, 3, 2, 1, 8, 12, 0, 3, 20, 5, 13, 14, 0, 15, 2, 20, 10, 9, 8, 7, 19, 5, 20, 15, 9, 15, 15, 13, 8, 11, 6, 14, 0, 4, 0, 15, 18, 19, 15, 3, 14, 20, 1, 8, 9, 12, 2,
16, 11, 19, 15, 14, 1, 2, 20, 3, 12, 0, 14, 1, 4, 17, 2, 12, 16, 6, 18, 3, 0, 12, 9], [7, 4, 4, 20, 12, 5, 17, 13, 17, 11, 6, 6, 18, 20, 15, 7, 11, 13, 13, 8, 19, 10, 4, 2, 9,
13, 2, 0, 8, 12, 1, 4, 7, 18, 3, 1, 4, 14, 8, 3, 19, 8, 4, 2, 1, 1, 13, 4, 17, 3, 1, 2, 14, 5, 12, 2, 17, 0, 19, 18, 12, 10, 8, 11, 19, 19, 3, 4, 6, 4, 13, 7, 17, 4, 15, 11, 0,
0, 13, 8, 16, 15, 6, 9, 7, 18, 11, 0, 0, 19, 15, 11, 11, 14, 12, 5, 9, 6, 15, 11], [0, 14, 8, 9, 0, 16, 0, 6, 9, 7, 2, 3, 1, 6, 15, 15, 6, 19, 16, 1, 5, 6, 19, 7, 7, 10, 11, 10,
2, 6, 20, 6, 17, 16, 2, 6, 10, 6, 14, 13, 0, 16, 5, 6, 4, 6, 10, 2, 9, 13, 10, 0, 13, 15, 17, 12, 16, 14, 14, 18, 5, 13, 14, 16, 2, 0, 5, 0, 5, 18, 0, 10, 3, 12, 5, 13, 13, 1,
11, 12, 17, 13, 10, 1, 1, 2, 13, 5, 7, 8, 18, 16, 7, 12, 19, 16, 0, 4, 8, 0], [1, 19, 1, 14, 5, 2, 12, 10, 17, 4, 14, 16, 1, 6, 5, 7, 15, 9, 0, 9, 9, 0, 8, 14, 12, 4, 8, 10, 4,
16, 0, 19, 10, 4, 3, 4, 12, 11, 12, 2, 14, 17, 16, 8, 18, 1, 10, 3, 1, 16, 19, 10, 3, 6, 12, 13, 10, 9, 14, 8, 8, 0, 16, 5, 1, 12, 18, 14, 0, 13, 3, 11, 20, 12, 18, 5, 16, 8,
6, 8, 17, 8, 4, 18, 0, 18, 1, 4, 8, 17, 12, 11, 15, 20, 6, 15, 2, 20, 12, 16], [8, 8, 7, 1, 14, 3, 18, 10, 4, 10, 8, 1, 11, 4, 14, 12, 11, 20, 16, 18, 3, 17, 16, 15, 2, 2, 9,
10, 8, 20, 8, 5, 7, 13, 2, 15, 18, 7, 20, 11, 8, 20, 12, 16, 8, 2, 7, 13, 3, 12, 18, 13, 6, 19, 13, 18, 1, 2, 10, 5, 20, 5, 11, 11, 20, 18, 7, 7, 11, 13, 14, 5, 9, 11, 18,
15, 10, 2, 15, 9, 3, 6, 2, 9, 18, 20, 11, 16, 19, 6, 20, 16, 15, 11, 7, 15, 16, 16, 19, 14], [11, 19, 14, 4, 3, 11, 8, 0, 6, 13, 1, 8, 1, 12, 3, 12, 10, 8, 4, 0, 19, 19, 1,
11, 18, 2, 3, 0, 4, 16, 7, 10, 18, 8, 7, 14, 7, 15, 9, 9, 14, 10, 0, 15, 5, 17, 19, 16, 10, 2, 8, 8, 16, 10, 16, 5, 9, 8, 17, 1, 17, 0, 18, 3, 3, 1, 5, 18, 7, 0, 7, 5, 11, 7, 5,
19, 17, 14, 4, 5, 11, 8, 19, 0, 17, 10, 20, 14, 15, 9, 20, 17, 11, 0, 20, 13, 8, 18, 20, 15], [11, 13, 6, 1, 5, 7, 6, 10, 13, 17, 0, 17, 15, 8, 9, 9, 0, 19, 14, 16, 6, 1, 8,
14, 1, 20, 0, 7, 3, 10, 1, 9, 10, 12, 11, 6, 10, 5, 5, 8, 10, 17, 9, 4, 5, 14, 1, 3, 0, 13, 2, 3, 15, 18, 19, 1, 12, 14, 18, 11, 9, 7, 14, 0, 20, 15, 2, 20, 20, 19, 2, 7, 17,
10, 11, 15, 0, 5, 16, 18, 3, 4, 12, 9, 6, 14, 8, 1, 5, 0, 15, 1, 0, 16, 20, 14, 7, 8, 15, 2], [12, 2, 0, 11, 3, 20, 9, 19, 3, 16, 8, 9, 18, 0, 12, 3, 6, 12, 0, 3, 3, 17, 3, 7,
17, 13, 20, 0, 5, 17, 20, 2, 10, 8, 9, 9, 6, 9, 1, 20, 1, 20, 1, 1, 16, 16, 15, 10, 11, 6, 9, 9, 1, 7, 14, 0, 15, 11, 2, 2, 8, 11, 18, 5, 2, 7, 7, 6, 13, 18, 12, 10, 1, 16, 8,
1, 6, 19, 12, 11, 4, 17, 16, 13, 13, 6, 3, 18, 2, 13, 9, 3, 8, 14, 14, 13, 15, 19, 10, 0], [1, 11, 12, 19, 11, 10, 19, 4, 6, 10, 18, 0, 19, 16, 7, 15, 7, 3, 16, 18, 1, 18, 8,
3, 11, 9, 6, 17, 9, 16, 18, 19, 16, 8, 17, 16, 12, 3, 19, 1, 5, 13, 18, 8, 18, 5, 4, 7, 1, 0, 4, 12, 8, 18, 20, 7, 5, 17, 2, 0, 6, 10, 13, 6, 13, 0, 18, 10, 8, 15, 5, 7, 16,
10, 3, 14, 4, 7, 5, 17, 16, 19, 20, 14, 16, 6, 15, 17, 3, 15, 18, 11, 4, 13, 2, 12, 9, 18, 20, 6]]