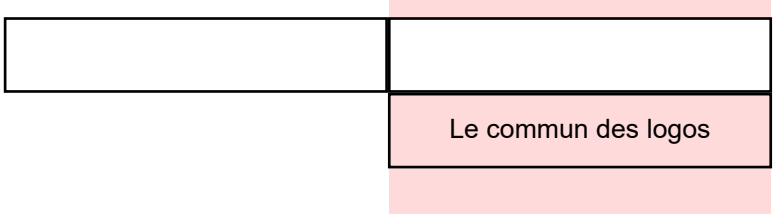


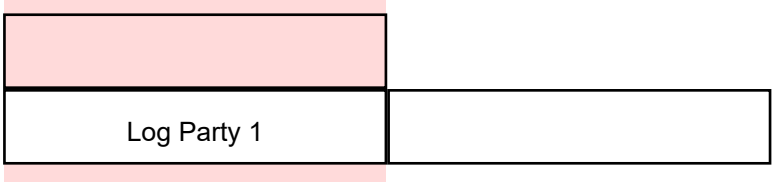

```
ipsup = []  # Descendance...
```

```
print('MusicAtoumic content')
```

```
nbr = 99988777666554433222111
```

```
dvinf = [372077943246799008798 1]
```

```
print('RUN unique, pour un problème commun')
```



logcomplix.py

```
#!/usr/bin/env python3.6
# -*- coding: utf-8 -*-
# MusicAtoumic logcomplix.py Ce 28 décembre 2017

""" RUN unique, pour un problème commun"""
print('MusicAtoumic content')

# Comment exploiter les tranches communes ?
""" Pour ce nombre :nbr:, qui développe des communs
En partant de la petite communauté (chiffre seul)
En croisant un diviseur commun d'ordre premier
    Premier à multiplier en tant que commun
    Première production et limite commune
        :dvinftable: Les Limites Communes
:oniveau: Les Communs Premiers
    Cet ensemble est une production conditionnée
    En réponse en terme de multiple commun :nbr:
    Chaque élément :oniveau: a son :dvinftable:
        :oniveau[1]: Premier commun
            :oniveau[1] = 3
        :dvinftable[0]: Première limite
            :dvinftable[0] = 37032917691353905345674893
"""  # Commencer une histoire communautaire
# Cet assemblage est une tenue de route
nbr = 99988877766655544433322111
dvinf = [3720779432467990087981]
ipsup = []  # Descendance...
# Le couple (ip&dv) multiplié, est commun si (:ip*dv = nbr:)
""" :nbr: Losqu'il a des tranches communes
:dvinftable: A des limites développées
    Si la limite de poids fort n'est pas d'ordre premier
    C'est qu'un couple commun produit sa reproduction
    Et, s'il elle est une limite n'ayant pas de diviseur
        Lorsqu'une série de diviseurs communs
        Forme une quantité de quotients (et multiples)
        Ici, (:oniveau:) produit:
            :nbr // dvinftable[2] = 268731:
            :oniveau = 3*37*269 = 268731:
            :oniveau * dvinftable[2] = nbr:
            Cette résolution est vraie:
                Tant que (:min(dvinftable):) EST PREMIER
    Chaque limite est ou n'est pas une Limite Première
:oniveau: A un élément ajouté:
    :dvinf[0]: Limite (:dvinftable[-1]:)
    Parvenir à cerner l'indivisible à l'aide du divisible
    Les points forts et faibles ont ce commun ajouté
    Toujours situé à proximité de la racine carrée (:nbr**.5:)
    Comme on peut le voir ci-dessous, au cas par cas
    À l'image d'un éventail circulant entre (:nbr&dvinf[0]:)
"""  # Répartition des communautés
print('**************')
print('nombre nbr:', nbr)
print('**************')
print('Quotient Premier:', dvinf[0])
print('**************')
"""
:cas: Unité ("IP") couplée au Quotient Premier ("DV")
:dvin: "ip.DV(fort)" Unité de couple ("DV")
```

```
:dvinf[0]: Nombre Premier connu ("DV") de poids faibe
    Ce nombre (:dvinf[0]:) est inconnu au début...
:cas = dvin // dvinf[0]: Et, "IP.dv(fort)"
    Le cas (:dvin:) est un élément de couple lié au
    Quotient Premier (":dvinf[0]:"). Atteignant aussi
    un niveau tranché pas encore développé, voire inconnu.
    En ascension vers les communs inconnus jusqu'ici
:but: Unité ("IP") assimilée en mode Quotient Premier
:ipsu: "IP.dv(fort)" Commun de poids fort
:cas: Unité ("IP") du Quotient Premier (":dvinf[0]:")
:but = ipsu * cas: Ainsi qu'("IP.dv(fort)")
    Le cas (:but:) a un but, celui d'associer une tranche
    aux nombres multiples. La série des (":ipsu:") unie
    le plus fort multiple(27) issu du Nombre Premier(3),
    avec le cas (:cas:) comme argument couplé.
        ("ipsuptable: [27, 999, 268731]")
        Composé d'unités de poids forts
            ("oniveau: [1, 3, 37, 269]")
            Composé d'unités de poids faibles
"""  # Propagation méthodique du couple
""":ipsuptable: Composition des maximums
:for dvin in dvinftable: Cas inférieurs à (:nbr:)
    Aux éléments "DV" couples diviseurs (:nbr:)
    Comme premiers quotients démultipliés (:cas:)
    :cas: Est le quotient (:dvin // dvinf[0]:)
        Et l'élément de couple commun à (:nbr:)
        Ce cas a une valeur "IP" utile
:for ipsu in ipsuptable: Premières tranches communes
    Aux éléments "IP" en couple "DV" d'(:dvinftable:)
    Comme une référence d'intervalle commun
    :but: Est (:ipsu(IP) * cas(IP):)
        :ipsu: Commun fort du premier commun
        ***Nombre Premier == Commun Premier
"""  # Nombres Premiers multiples (:nbr:)
"""
Le premier nombre multiple (:nbr:) Produit des communs
Il forme une communauté de nombres d'éléments de poids ("IP")
Cette première tranche occupe une quantité de diviseurs, à la
limite du niveau de seuil hors du commun. L'élément ("IP") de
poids fort ayant peu de communs, apporte tant de réponses. Ici,
le cas ("DV") a des arguments:
    IP.dv(faible) = 3 | ip.DV(faible) = 333296259222185148111074037
    IP.dv(fort) = 27 | ip.DV(fort) = 37032917691353905345674893
    ip.DV(fort): Premier échelon de poids fort, dans (:dvinftable:)
Ici, le Nombre Premier ("DV") est connu pour sa démultiplication.
Il démultiplie les communs spécifiquement relevés, trouvant leurs
racines le Nombre Premier ("DV"), proche d'(:ip.DV(fort)**.5:).
"""  # Mise en condition thématique
"""
*:dvinftable[0]: Tranche n° 1
Premiers ( 4 )ex [1, 37, 269]
1 * 37032917691353905345674893 Types 1&5
37 * 10008896673338893336668889 Types 1&5
269 * 137668839001315633255297 Types 5&1
9953 * 3720779432467990087981 Types 5&1
nphullmen_7x.py En: 0.37087273597717285

*:dvinftable[1]: Tranche n° 2
Premiers ( 2 )ex [1, 269]
1 * 10008896673338893336668889 Types 1&5
269 * 3720779432467990087981 Types 5&1
```

```
nphullmen_7x.py En: 0.24109673500061035

*Comparaison Cosmic
Premiers ( 16 )ex [1, 3, 37, 269]
1 * 999888777666555444333222111 Types 1&3
3 * 333296259222185148111074037 Types 3&3
9 * 111098753074061716037024679 Types 3&3
27 * 37032917691353905345674893 Types 3&5
37 * 270240210180150120090060003 Types 1&3
111 * 900800700600500400300200 1 Types 3&3
269 * 37170586530355220978930 19 Types 5&3
333 * 30026690020016680010006 67 Types 3&3
807 * 12390195510118406992976 73 Types 3&3
999 * 10008896673338893336668 89 Types 3&5
2421 * 413006517003946899765891 Types 3&3
7263 * 137668839001315633255297 Types 3&1
9953 * 100461044676635732375487 Types 5&3
29859 * 33487014892211910791829 Types 3&3
89577 * 11162338297403970263943 Types 3&3
268731 * 3720779432467990087981 Types 3&1
nphullmen_7x.py En: 0.6955530643463135
"""  # Cas par cas des tranches composées
print('RUN unique, pour un problème commun')

```