DM : Logarithme discret

L'objectif de ce DM est d'implanter l'algorithme de logarithme discret de Pohlig-Hellman dans le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ avec $p = 71166625531 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 + 1$, en prenant $2 \bmod p$ comme générateur.
Le fichier `entier_modulaire.py` fournit des utilitaires pour manipuler des entiers modulaires. On vous renvoie au DM2 pour les opérations de bases dans la classe `Zp`.

Consignes.

- Modifier le fichier `DM3.py` en fournissant toutes les implantations demandées. Ne pas modifier le fichier `entier_modulaire.py`, ni les signatures des fonctions dans `DM3.py`.
- **Rendu.** Vous devez rendre le fichier `DM3.py` à l'emplacement prévu sur Moodle.
- Pour les questions qui ne demandent pas de code, vous pouvez répondre en commentaire dans `DM3.py`.

Exercice 1. *Algorithme des restes chinois*

Pour obtenir un algorithme qui reconstruit $(x \bmod q)$ à partir des $x_i := (x \bmod q_i)$, nous allons nous inspirer de son équivalent polynomial : l'interpolation de Lagrange. Ici, $q = \prod_{i=1}^{k} q_i$ et les q_i sont premiers entre eux.

1. Proposer un entier M_i dont les résidus sont $M_i \bmod q_j = 0$ si $i \neq j$ et $M_i \bmod q_i \neq 0$.
2. À partir de M_i et u_i, proposer un entier $0 \leq v_i < q_i$ tel que $(M_i v_i) = u_i \bmod q_i$. Pourquoi doit-on supposer que les q_i sont premiers entre eux ?
   Notez que l'on a toujours $(M_i v_i) = 0 \bmod q_j$ pour $j \neq i$.
3. Trouver l'unique $0 \leq x < q$ tel que $x = x_i \bmod q_i$ pour tout i.
   Aide : *Si vous avez un candidat x' qui satisfait $x' = x_i \bmod q_i$ mais est potentiellement plus grand que q, il suffit de le réduire modulo q.*
4. Compléter l'algorithme `CRT(residues, moduli)`.

Exercice 2. *Pohlig-Hellman*

1. Compléter la fonction `LD(h,g)` qui calcule naïvement le logarithme discret de h en base g.
2. Compléter l'algorithme `LD_PH(h,g,qs)` qui calcule le logarithme discret par la méthode de Pohlig-Hellman en se réduisant à des calculs de logarithme discret naïf sur des instances plus petites.
   Note : *Si votre fonction `CRT` ne marche pas, renvoyez juste la liste des $(x \bmod q_i)$.*

Tests et exemples.
Voici quelques exemples qui montre le fonctionnement souhaité des différentes fonctions et qui peuvent vous servir de test.

```
>>> import math
>>> qs = [2, 3, 5, 7, 11**2, 13, 17, 19, 23, 29]
>>> p = math.prod(qs)+1
>>> g = Zp(2,p) # Générateur modulo p
>>> h = Zp(4975429640,p)
>>> LD(h,g)
48472
>>> h = Zp(16002076369,p)
>>> LD_PH(h,g,qs)
63895244811
>>> CRT([1,0,1,2],[2,3,5,7])
51
>>> # Autres 'rappels'
>>> 15//3        # Division exacte -> renvoie un entier
>>> sum([1,2,3]) # somme d'une liste
```