Corrigé : sécurité par mot de passe

Partie I. Base de données

Question 1.

```sql
SELECT Idclient FROM client WHERE Pays = 'CH' ;
```

Question 2.

```sql
SELECT Idproduit, COUNT(*) FROM vente GROUP BY Idproduit ;
```

Question 3.

```sql
SELECT Mail, SUM(Prix) as Total FROM client JOIN vente ON Idclient = Client
  WHERE Annee = 2024
  GROUP BY Idclient
  ORDER BY Total DESC LIMIT 10 ;
```

Question 4.

```sql
SELECT Mail, SUM(Prix) as Total FROM client JOIN vente ON Idclient = Client
  WHERE Annee = 2024
  GROUP BY Idclient HAVING Total > 100 ;
```

Question 5.

```sql
SELECT Idclient FROM client
  WHERE Idclient NOT IN (SELECT Client FROM vente WHERE Annee = 2024) ;
```

Question 6.

```sql
SELECT c1.Idclient, c2.Idclient FROM client AS c1 JOIN client AS c2 ON c1.Mail = c2.Mail
  WHERE c1.Idclient < c2.Idclient ;
```

Partie II. Conversion de données

Question 7.

```python
def str2int(s):
    somme = 0
    for k in range(len(s)):
        somme += ord(s[k]) * 128 ** k
    return somme
```

Question 8. On observe que si $x = \sum_{k=0}^{n-1} \varphi(c_k) \times 128^k$ alors $x \bmod 128 = \varphi(c_0)$ et $\lfloor x/128 \rfloor = \sum_{k=0}^{n-2} \varphi(c_{k+1}) \times 128^k$. D'où la fonction :

```
def int2str(x):
    s = ''
    while x > 0:
        s = s + chr(x % 128)
        x = x // 128
    return s
```

Question 9.

```
def conforme(mdp):
    if len(mdp) < 12:
        return False
    upp, low, num, spe = False, False, False, False
    for c in mdp:
        if 65 <= ord(c) <= 90:
            upp = True
        if 97 <= ord(c) <= 122:
            low = True
        if 48 <= ord(c) <= 57:
            num = True
        if 33 <= ord(c) <= 47:
            spe = True
    return upp and low and num and spe
```

Partie III. Une fonction de hachage cryptographique

Question 10. Stocker en clair les mots de passe permettrait à toute personne pouvant accéder à cette base de donnée de récupérer les identifiants et mots de passe des clients et donc d'effectuer des achats en se faisant passer pour eux.

Question 11. Une fonction de hachage cryptographique doit être facile à calculer, mais il doit être (autant que faire se peut) impossible de déterminer les antécédents d'une valeur donnée.

Plus précisément, une fonction de hachage cryptographique idéale possède les six propriétés suivantes :

- la fonction est déterministe, c'est-à-dire qu'une même chaîne de caractères aura toujours la même valeur de hachage ;
- la valeur de hachage d'un message se calcule « facilement » ;
- il est impossible, pour une valeur de hachage donnée, de construire une chaîne de caractères ayant cette valeur ;
- il est impossible de trouver une seconde chaîne de caractère ayant la même valeur de hachage qu'une chaîne de caractère donnée ;
- il est impossible de trouver deux chaînes de caractères différentes ayant la même valeur de hachage ;
- modifier un tant soit peu la chaîne de caractère modifie considérablement la valeur de hachage.

Question 12.

```
def encrypte(s):
    z = str2int(s)
    x = z % (2 ** 64)
    y = z // (2 ** 64)
    H = 0
    for k in range(128):
        H += (x % 2) * 2 ** k
        x = (((2 * y + 3) * x) // 2) % (2 ** 128)
    return H
```