Sujet 2

EXERCICE 1 (10 points)

Programmer une fonction `renverse`, prenant en paramètre une chaîne de caractères non vide `mot` et renvoie cette chaîne de caractères en ordre inverse.

Exemple :

```
>>> renverse("")
""
>>> renverse("abc")
"cba"
>>> renverse("informatique")
"euqitamrofni"
```

EXERCICE 2 (10 points)

On affecte à chaque lettre de l'alphabet un code selon le tableau ci-dessous :

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
1	2	3	4	5	6	7	8	9	10	11	12	13
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
14	15	16	17	18	19	20	21	22	23	24	25	26

Cette table de correspondance est stockée dans un dictionnaire `dico` où les clés sont les lettres de l'alphabet et les valeurs les codes correspondants.

```
dico = {
    "A": 1, "B": 2, "C": 3, "D": 4, "E": 5, "F": 6,
    "G": 7, "H": 8, "I": 9, "J": 10, "K": 11, "L": 12, "M": 13,
    "N": 14, "O": 15, "P": 16, "Q": 17, "R": 18, "S": 19, "T": 20,
    "U": 21, "V": 22, "W": 23, "X": 24, "Y": 25, "Z": 26
}
```

Pour un mot donné, on détermine d'une part son **code alphabétique concaténé**, obtenu par la juxtaposition des codes de chacun de ses caractères, et d'autre part, son **code additionné**, qui est la somme des codes de chacun de ses caractères.

Par ailleurs, on dit que ce mot est *parfait* si **le code additionné divise le code concaténé**.

Exemples :

- Pour le mot `"PAUL"`, le code concaténé est la chaîne `"1612112"`, soit l'entier `1 612 112`.
  Son code additionné est l'entier `50` car `16 + 1 + 21 + 12 = 50`.
  `50` ne divise pas `1 612 112`. Ainsi, le mot `"PAUL"` *n'est pas parfait*.
- Pour le mot `"ALAIN"`, le code concaténé est la chaîne `"1121914"`, soit l'entier `1 121 914`.

Le code additionné est l'entier `37` car `1 + 12 + 1 + 9 + 14 = 37`.
`37` divise `1 121 914`. Ainsi, le mot `"ALAIN"` *est parfait*.

À compléter :

Compléter la fonction `codes_parfait` ci-dessous qui prend en paramètre un mot en majuscule et renvoie un triplet contenant le **code additionné**, le **code concaténé** (sous forme d'entier) et un **booléen** indiquant si le mot est parfait ou non.

```
def codes_parfait(mot):
    """Renvoie un triplet
    (code_additionne, code_concatené, mot_est_parfait) où :
    - code_additionne est la somme des codes des lettres du mot ;
    - code_concatené est le nombre formé par la concaténation des codes ;
    - mot_est_parfait est un booléen indiquant si le mot est parfait.
    """
    code_concatenate = ""
    code_additionne = 0
    for c in mot:
        code_concatenate = code_concatenate + ...
        code_additionne = code_additionne + ...
    code_concatenate = int(code_concatenate)
    mot_est_parfait = ...
    return code_additionne, code_concatenate, mot_est_parfait
```

Exemples attendus :

```
>>> codes_parfait("PAUL")
(50, 1612112, False)

>>> codes_parfait("ALAIN")
(37, 1121914, True)
```