```
from doctest import testmod


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}


def est_parfait(mot) :
    """Renvoie les codes concaténé et additionné d'une chaîne de caractère ainsi qu'
un booléen indiquant si le mot est parfait ou pas.

    Args:
        mot (str): mot à analyser

    Returns:
        list:   * int: code concaténé de mot
                * int: code additionné de mo
                * bool: est ce que le mot est parfait?

    Tests et Exemples:
    >>> est_parfait("PAUL")
    [50, 1612112, False]
    >>> est_parfait("ALAIN")
    [37, 1121914, True]
    """
    #mot est une chaîne de caractères (en lettres majuscules)
    code_c = ""

    # initialisation du code additionné avec la valeur 0
    code_a = 0

    # parcours caractère par caractère
    for c in mot :
        # concaténation de code_c avec le code du
        # caractère courant c
        code_c = code_c + str(dico[c])

        # ajout au code additionné la valeur de code
        # du caractère courant c
        code_a = code_a + dico[c]

    code_c = int(code_c)

    # un mot est parfait si le code concaténé est
    # divisible par le code additionné
    # càd : si le reste de la division euclidienne
    # vaut bien 0
    if code_c % code_a == 0 :
        mot_est_parfait = True
    else :
        mot_est_parfait = False
    return [code_a, code_c, mot_est_parfait]


assert est_parfait("PAUL") == [50, 1612112, False]
assert est_parfait("ALAIN") == [37, 1121914, True]
testmod()
```