3 - MPI (4 pts) – *Answer on copy n°2*

We try to decrypt a long message "**mess**" in parallel. We have a "**secret**" key of the same length as the message composed only of lowercase letters. The **secret** allows us to know which characters of the message "**mess**" are to be kept. We search in "**secret**" the most present letter (only lower case letters) and we keep only the letters in "**mess**" at these positions.

- **secret**: 'abaadzaga'
- **mess**: 'mipiza!a!'
- We keep only '**mpi!!**' as in **secret** 'a' is the most present letter and '**a**b**aa**dz**a**g**a**' donne les positions '**m**i**pi**za**!**a**!**'

To verify that the message has not been modified, a signature is calculated. We count the number of characters present only once in the decrypted message. In our case the signature is 3 because 'm', 'p' and 'i' are present only once while '!' is present twice.

Parallelize the following code by reading (**read_secret** and **read_encrypted** functions) and displaying only on the rank 0 process. Consider having all the provided functions directly usable.

Provided functions:

- `read_secret()` : reads the secret key
- `read_encrypted()` : reads the encrypted text
- `add_list(l1, l2)` : adds two lists, elements by elements `add_lists([1,2], [3,4])` returns [4,6]
- `count_letters(text)` : counts for each letter the number of times it appears in "`text`". Returns an array "`result`" of 26 integers. `result[0]` is the number of 'a' in "`text`"
- `get_max_letter(lst)` : returns the letters associated with the maximum number of occurrences in a 26 elements list. The argument is an array of 26 elements (such as one returned by count_letters). If the maximum is in cell 3, returns 'd'; if the maximum is in cell 0 returns 'a'
- `keep_only(secret, document, letter)` : keeps only the characters from documents aligned with the letter "`letter`" in `secret`. `keep_only('abaadzaga', 'mipiza!a!', 'a')` returns `'mpi!!'`
- `get_signature(text, binf, bsup)` : gets the signature of a text between `binf` and `bsup` by looking at the whole text. `get_signature('mpi!!', 1, 4)` returns 2 by looking at `'pi!'` where only `'p'` and `'i'` are present one time in `'mpi!!'`.

Sequential version (`keep_only` and `get_signature` are only presented for better understanding):

```
def keep_only(secret, document, letter):
    res = ''
    for i in range(len(secret)):
        if secret[i] == letter:
            res += document[i]
    return res


def get_signature(text, binf, bsup):
    nb = 0
    for pos in range(binf, bsup):
        if text.count(text[pos]) == 1:
            nb += 1
    return nb

secret = read_secret()
document = read_encrypted()
# Returns an array of 26 numbers
count = count_letters(secret)

# Get the letter with the highest count
max_letter = get_max_letter(count)

# Uncrypt the document using the secret
# and the max letter
uncrypted = keep_only(secret, document,
max_letter)

# Checks the number of character present
# only once in the uncrypted text
signature = get_signature(uncrypted, 0,
len(uncrypted))

print(uncrypted, signature)
```