TD 6 : Prog.

```
def mystere(l:list[int]) -> int:
    """
    precondition: non vide, contient des elts positifs ou nuls
    """
    res = 0
    for elt in l:
        if elt > res:
            res = elt
    return res
```

1. Dérouler avec un tableau de la mémoire l'exécution de `mystere([4, 3, 4, 6])`.
2. Compléter la documentation de cette fonction.

1.

Res	elt	L
0	/	[4,3,4,6]
0	4	"
4	3	"
"	4	"
"	6	"
6	/	"

2. Exemples :

$$$ mystere([2,6,11,3])
11
$$$ mystere([7,2,1,9])
9.
$$$ mystère([12,7,1,3])
12.

Renvoie le plus grand élément de L.

3. Une fonction `nombre_occurrences` qui prend en paramètre un caractère et une chaîne `chaine` et qui renvoie le nombre d'occurrences de `caracteres` dans `chaine` :

```
$$$ nombre_occurrences('a', 'abracadabra')
5
```

```
def nombre_occurences(char:str, chaine:str)->int:
    '''
    Renvoie le nombre d'occurences de char dans chaine
    Précondition: len(char)==1.
    Exemples:
    $$$ nombre_occurences('e', 'element'):
    3
    $$$ nombre_occurences('a', 'abracadabra')
    5.
    '''
    ret=0
    for lettre in chaine:
        if lettre == char:
            ret+=1.
    Return ret.
```

4. Une fonction `nombre_occurrences2` qui prend en paramètre une chaine `caracteres` non vide et une chaîne `chaine` et qui renvoie le nombre d'occurrences cumulé des caractères de `caracteres` dans `chaine` :

```
$$$ nombre_occurrences2('afbcg', 'abracadabra')
8
```

```
def nombre_occurences2(caracteres:str, chaine:str)->int:
    '''
    Renvoie le nombre d'occurences cumulé des caractères dans chaine
    Préconditions : /
    Exemples :
    $$$ nombre_occurences2('afbcg', 'abracadabra')
     8
    $$$ nombre_occurences2('au', 'salut')
     2.
    '''
    Ret = 0.
    for el in caracteres:
        Ret += nombre_occurences(el, chaine)
    Return Ret.
```

5. Une fonction `produit` qui prend en paramètre une liste d'entiers et qui renvoie le produit des éléments qu'elle contient.

```
$$$ produit([2, 3, -5])
-30
```

```
def produit(tab: list[int]) -> int
    """
    Renvoie le produit des éléments de tab.
    Préconditions : len(tab) >= 1.
    Exemples :
    $$$ produit([1,2,3])
    6
    $$$ produits([6, 5, 2, 3])
    180
    """
    ret = 1
    for elt in tab:
        ret *= elt
    return ret.
```

6. Une fonction `pairs_et_positifs` qui prend en paramètre une liste d'entiers `l` et renvoie une nouvelle liste contenant uniquement les entiers strictement positifs et pairs de `l`, dans le même ordre. Que faudrait-il changer pour obtenir à la fois les entiers pairs et les entiers strictement positifs ?

```
$$$ pairs_positifs([2, 3, -6, 4, 5, 12])
[2, 4, 12]
```

```
def pairs_et_positifs(tab:list[int])->list[int]:
    """
    Renvoie la liste tab allégée de ses composantes
    impaires et négatives.
    Précond : /
    Exemple :
    $$$ pairs_et_positif([2,3,-6,4,5,12])
    [2,4,12].
    """
    ret=[]
    for elt in tab:
        if elt%2==0 and elt>0:
            ret.append(elt)
    return ret.
```

7. Écrire une fonction `incremente_chiffres` qui prend en paramètre une chaîne de caractères `chaine` et renvoie une chaîne contenant dans le même ordre les caractères de `chaine`, mais tels que les chiffres ont été remplacés par leur valeur incrémentée. Par exemple :

```
$$$ incremente_chiffres('3%$6')
'4%$7'
```

On utilisera `str.isdigit()`.

```
>>> 'a'.isdigit()
False
>>> '42'.isdigit()
True
```

```
def incremente_chiffres(chaine:str)->str:
    '''
    Précond: /
    Exemples:
    $$$ incremente-chiffres('3%$6')
    '4%$7'
    '''
    Ret=''
    for elt in chaine:
        if elt.isdigit:
            ret+=str(int(elt)+1)
        else:
            ret+=elt.
    Return Ret.
```