

TP OOP N°2

Master 1 MISC

École d'ingénieurs du Littoral-Côte-d'Opale

Université du Littoral Côte d'Opale

Document à rendre :

un rapport de TP qui contient

les commandes utilisées,

avec description et références à l'énoncé,

et commentaires éventuels.

**Attention**

| Lisez ATTENTIVEMENT les explications et les consignes de travail.

**Attention**

| Remarque : vous conserverez une trace numérique de toutes les actions réalisées dans le fichier de scripts. Cela vous permettra de relancer des ensembles de commandes en une seule fois. Vous remettrez le fichier à la fin de chaque séance.

-
1. Le factoriel d'un entier positif n est : $n! = \prod_{j=1}^n j$ avec $0! = 1$.
Écrivez un code qui :
 - (a) Définit une fonction `factoriel(n)` qui calcule et renvoie $n!$ (avec n supposé positif ou nul).
 - (b) Saisit un entier positif puis calcule et affiche la table des factoriels des nombres j entre 0 et l'entier saisi.
 - (c) Combien de chiffres faut-il pour écrire l'entier $50!$? (Résultat: 65 chiffres)

Jeux de tests :

10 ==> 1 2 6 24 120 720 5040 40320 362880 3628800

2. Écrivez un code qui définit une fonction `icarre(n)` qui calcule et renvoie le carré d'un entier positif n par la méthode des impairs, laquelle consiste à faire la **somme** des n premiers **nombre**s **impairs**.
 - $1^2=1$: addition du premier impair
 - $2^2=1+3=4$: addition des deux premiers impairs
 - $3^2=1+3+5=9$: addition des trois premiers impairs
 - $4^2=1+3+5+7=16$: addition des quatre premiers impairs
 - $5^2=1+3+5+7+9=25$: addition des cinq premiers impairs
 - Et ainsi de suite
3. Écrivez un code qui :
 - (a) Définit une fonction `mult0(a,b)` qui calcule et renvoie le produit de deux entiers positifs a et b sans faire appel à l'opérateur de multiplication.
 - (b) Déduisez une fonction `mult(a,b)` qui calcule et renvoie le produit de deux entiers a et b sans faire appel aux fonctions prédéfinies (signe ou valeur-absolue -- vous devez les redéfinir).
 - (c) Saisissez deux entiers puis affichez le résultat.

Jeux de tests :

10 5 ==> 50 -50 50
-21 -123 ==> 2583 -2583 2583

4. Écrivez un code qui saisit un mot, transforme ce mot en liste de caractères, puis transforme cette liste en chaîne en insérant un espace entre chaque caractère et affiche la chaîne.

Jeux de tests :

altitude ==> a l t i t u d e
et123 ==> e t 1 2 3

5. Écrivez un code qui définit une fonction `noccurs(lt,e)` qui calcule et renvoie le nombre de fois qu'un élément `e` apparaît dans une liste `lt`, et `-1` si `lt` n'a pas le bon type.

Jeux de tests :

```
noccurs(["a",1,2,3,2,2,5,2],2) ==> 4
noccurs(2,2) ==> -1
noccurs(["a",1,2,3,2,2,5,2],9) ==> 0
```

6. Écrivez un code qui saisit un entier dans `n` et qui génère la liste : `["$-$","$--$","$---$",...]`

Le dernier élément possède `n` tirets.

Jeux de tests :

```
5 ==> ['$-$', '$--$', '$---$', '$----$', '$-----$']
0 ==> []
-2 ==> []
1 ==> ['$-$']
```

7. Écrivez un code qui définit une fonction `traceMT(m)` qui calcule et renvoie la trace d'une matrice numérique (int ou float) `m` de taille `n×n` des valeurs :

$\text{trace}(M) = \sum_{j=1..n} M_{jj}$

c.-à-d. la somme des valeurs de la première diagonale de la matrice. Exemple : La trace de la matrice ci-dessous est 15 (somme des valeurs 1, 5 et 9 qui forment la première diagonale) :

```
1 2 3
4 5 6
7 8 9
```