```
// cette fonction récursive renvoie
// un int (fact) correspondant au résultat de l'opération
// factorielle du nombre passé en paramètre (nb)
// si l'argument n'est pas valide
// (nombre inférieur à 0), la fonction doit renvoyer 0
// REMARQUE : cette fonction ne gère pas les int overflow

// EXEMPLE :
// la factorielle de 5 (notée 5!) est égale à :
// 5 * 4 * 3 * 2 * 1
// ce qui équivaut à
// 5 * 4 * 3 * 2

int     ft_recursive_factorial(int nb)
{
        // si nb n'est pas valide (si nb < 0)
        if (nb < 0)
                // on retourne 0
                return (0);

        // CAS DE BASE DE LA RECURSION :
        // (condition de sortie de la récursivité)
        // 0! = 1
        // 1! = 1

        // sinon, si nb est inférieur ou égal à 1
        // (donc si nb est égal à 0 ou à 1)
        else if (nb <= 1)
                // on retourne 1
                return (1);

        // si nb est supérieur à 1
        // (si on est pas déjá sorti de la fonction car nb est inférieur ou égal à 1)
        // la fonction s'appelle elle-même avec l'argument nb - 1
        // jusqu'à ce que nb atteigne 1 (CAS DE BASE DE LA RECURSIVITE)
        // ainsi, à chaque étape, nb est multipliée par
        // la factorielle du nombre immédiatement inférieur
```

```
// EXEMPLE AVEC nb = 5
// 5! = 5 * 4 * 3 * 2
// OR 4! = 4 * 3 * 2
// DONC 5! = 5 * 4!

// 4! = 4 * 3 * 2 * 1
// OR 3! = 3 * 2 * 1
// DONC 4! = 4 * 3!

// 3! = 3 * 2 * 1
// OR 2! = 2 * 1
// DONC 3! = 3 * 2!

// 2! = 2 * 1
// OR 1! = 1
// DONC 2! = 2 * 1!

// DONC
// 5! = 5 * 4!
// 4! = 4 * 3!
// 3! = 3 * 2!
// 2! = 2 * 1!
// 1! = 1

// APPLICATION :
// nb = 5
// 1) 5 * ft_recursive_factorial(4) =>
// 2) 4 * ft_recursive_factorial(3) =>
// 3) 3 * ft_recursive_factorial(2) =>
// 4) 2 * ft_recursive_factorial(1) =>
// 5) ft_recursive_factorial(1) => 1
// on remonte dans les appels récursifs
// 4) 2 * ft_recursive_factorial(1) = 2 * 1 = 2 =>
// 3) 3 * ft_recursive_factorial(2) = 3 * 2 = 6 =>
// 2) 4 * ft_recursive_factorial(3) = 4 * 6 = 24 =>
// 1) 5 * ft_recursive_factorial(4) = 5 * 24 = 120

nb = nb * ft_recursive_factorial(nb - 1);
```

```
        // on retourne le résultat
        return (nb);
}

// RESULTAT : 120
#include "ft_putnbr.h"
#include "ft_recursive_factorial.h"

int     main(void)
{
        int     nb;
        int     fact;

        nb = 5;
        fact = ft_recursive_factorial(nb);
        ft_putnbr(fact);
        return (0);
}
```