```
// la fonction ft_ultimate_ft prend en argument
// un pointeur (*) vers un pointeur (*) vers un pointeur (*)
// vers un pointeur (*) vers un pointeur (*) vers un pointeur (*)
// vers un pointeur (*) vers un pointeur (*) vers un pointeur (*)
// vers un entier (int)
// ce pointeur est appelé nbr
// cet argument est donc une suite de 9 pointeurs chainés :
// ********* nbr

// il pointe vers l'adresse de l'adresse de l'adresse
// de l'adresse de l'adresse de l'adresse de l'adresse
// de l'adresse de l'adresse de la variable de type int
// passée en paramètre
// ATTENTION : c'est bien l'adresse finale qui doit être
// passée en paramètre, et non pas directement la variable
void ft_ultimate_ft(int *********nbr)
{
        // ATTENTION : ici, *********nbr est une "lvalue"
        // car elle se trouve à gauche (left) de l'opérande
        // d'affectation =
        // ********* sont donc ici des opérateurs de déréférencement chainés
        // cela signifie :
        // valeur située à l'adresse de l'adresse de l'adresse
        // de l'adresse de l'adresse de l'adresse
        // de l'adresse de l'adresse de l'adresse
        // fournie par le pointeur nbr
        // cette instruction modifie la valeur de la variable pointée par nbr
        // (la valeur de la variable se trouvant à l'adresse contenue dans *********nbr,
        // donc à l'adresse passée en paramètre de la fonction)
        // en lui assignant la valeur 42
        *********nbr = 42;
}

// MAIN v1 (ATTENTION : ne passe pas la norminette, voir plus bas pour la v2 à la norme) :
#include "ft_ultimate_ft.h"

// on inclut le fichier d'entête de la fonction externe ft_putnbr
#include "ft_putnbr.h"
```

```
int main (void)
{
        // on définit une variable de type int appelée n
        int n;

        // on définit un pointeur appelé pt1
        int *pt1;

        // on définit un pointeur de pointeur appelé pt2
        int **pt2;

        // on définit un pointeur de pointeur de pointeur appelé pt3
        int ***pt3;

        // etc.
        int ****pt4;
        int *****pt5;
        int ******pt6;
        int *******pt7;
        int ********pt8;

        // on fait pointer le pointeur pt1 vers l'adresse de la variable n)
        pt1 = &n;

        // on fait pointer le pointeur de pointeur pt2 vers l'adresse du pointeur pt1
        pt2 = &pt1;

        // on fait pointer le pointeur de pointeur de pointeur pt3 vers
        // l'adresse du pointeur de pointeur pt2
        pt3 = &pt2;

        // etc.
        pt4 = &pt3;
        pt5 = &pt4;
        pt6 = &pt5;
        pt7 = &pt6;
        pt8 = &pt7;
```

```
        // on exécute ft_ultimate_ft en lui passant l'adresse
        // de pt8 (le pointeur de 8ème niveau)
        // & signifie "adresse de "
        // on atteindra ainsi le 9eme niveau de pointeurs
        ft_ultimate_ft(&pt8);

        // pour afficher la valeur de la variable
        // pointée par le pointeur de pointeur de pointeur ... pt8
        // (donc la valeur de n)
        ft_putnbr(********pt8);

        // pour indiquer au système d'exploitation que le programme s'est terminé avec succès
        return (0);
}
```

MAIN v2 (passe à la norminette) :

```
// pour utiliser malloc et free
#include <stdlib.h>

#include "ft_ultimate_ft.h"
#include "ft_putnbr.h"

int     main(void)
{
        // variable de type int
        int     n;

        // pointeur de pointeur de pointeur de pointeur de pointeur
        // de pointeur de pointeur de pointeur (de "niveau 8")
        // ATTENTION : ft_ultimate_ft attend un pointeur de niveau 9
        // il faudra donc lui passer l'adresse de ce pointeur ptr
        int     ********ptr;

        // on alloue à ptr, pointeur de niveau 8,
        // la taille d'un pointeur sur un entier de niveau 7
        ptr = malloc(sizeof(int *******));
```

```
// on alloue à l'adresse du pointeur ptr de niveau 8,
// qui est ainsi de niveau 7,
// la taille d'un pointeur sur un entier de niveau 6
*ptr = malloc(sizeof(int ******));

// on alloue à l'adresse de l'adresse du pointeur ptr de niveau 8,
// qui est ainsi de niveau 6,
// la taille d'un pointeur sur un entier de niveau 5
**ptr = malloc(sizeof(int *****));

// etc
***ptr = malloc(sizeof(int ****));
****ptr = malloc(sizeof(int ***));
*****ptr = malloc(sizeof(int **));
******ptr = malloc(sizeof(int *));

// on alloue l'adresse de l'adresse de l'adresse
// de l'adresse de l'adresse de l'adresse de l'adresse
// du pointeur ptr de niveau 8
// qui est ainsi de niveau 1,
// l'adresse de la variable n
*******ptr = &n;

// on passe à ft_ultimate_ft l'adresse du pointeur de niveau 8
// donc un pointeur de niveau 9
ft_ultimate_ft(&ptr);

// on affiche la valeur de la valeur de la valeur ...
// de ptr
// qui est au final la valeur de n
// (la valeur à l'adresse de n,
// ********ptr pointant sur l'adresse de n)
// grâce à *******ptr = &n;
ft_putnbr(********ptr);

// on libère la mémoire allouée
// par malloc (du niveau 1 au niveau 7)
```

```
        free(*******ptr);
        free(******ptr);
        free(*****ptr);
        free(****ptr);
        free(***ptr);
        free(**ptr);
        free(*ptr);
        free(ptr);

        // on retourne 0 pour indiquer
        // que le programme s'est terminé sans erreur
        return (0);
}
```