

```

#include <unistd.h>

void    ft_print_combn(int n);
void    ft_recursive_combn(int *comb, int n, int pos, int prev);
void    ft_write_combn(int *comb, int n);
void    ft_write_number(int n);

// initialise le processus de génération des combinaisons
// prend comme paramètre n, de type int
// qui correspond aux n chiffres de la combinaison
// à générer
// par exemple, avec n = 2
// on génèrera des combinaisons de 2 chiffres

// CETTE FONCTION NE SERA EXECUTEE QU'UNE SEULE FOIS
// pour :
// - déclarer le tableau de combinaisons (comb)
// - initialiser pos avec la valeur 0
// - initialiser prev avec la valeur -1
// (voir plus bas)
void    ft_print_combn(int n)
{
    // on déclare un tableau pouvant contenir 9 éléments
    // de type int
    // comb[0] fera référence au premier élément
    // comb[1] fera référence au deuxième élément
    // comb[8] fera référence au neuvième (et dernier) élément

    // on choisit une taille de 9 car cela correspond
    // au nombre maximum de chiffres pouvant être utilisés
    // pour créer des combinaisons uniques sans répétition
    // avec n entre 1 et 9
    // (de 1 à 9 chiffres, selon la valeur de n)
    // on pourra donc générer des combinaisons de 9 chiffres maximum
    // (si n == 9)

    // chaque élément représentera un chiffre de la combinaison
    // en cours de construction

```

```

// par exemple, avec (n == 3)
// comb[0], comb[1] et comb[2] seront utilisés
// pour stocker au fur et à mesure chaque combinaison
// (012, puis 013, puis 014, etc jusqu'à 789)
// (voir plus bas)
int    comb[9];

// si 0 < n < 10 (comme indiqué dans le sujet)
// (condition pour que le programme continue)
if (n > 0 && n < 10)
{
    // on exécute ft_recursive_combn
    // pour générer chaque combinaison de n chiffres

    // cette fonction prend en paramètre

    // le tableau comb (vide au départ)
    // ATTENTION :
    // le tableau est en fait initialisé
    // avec des valeurs aléatoires !

    // le nombre de chiffres voulus pour la combinaison
    // (n)

    // la valeur 0
    // correspondant à pos
    // (VOIR LE PROTOTYPE DE LA FONCTION ft_recursive_combn)

    // la valeur -1
    // correspondant à prev
    // (VOIR LE PROTOTYPE DE LA FONCTION ft_recursive_combn)
    ft_recursive_combn(comb, n, 0, -1);
}

```

```

}

```

```

// APRES LA PREMIERE EXECUTION DE ft_recursive_combn
// comb[0] vaudra 0
// comb[1] vaudra 1

```

```

// comb[2] vaudra 2

// puis ft_write_combn écrira la valeur de comb[0] (0)
// puis celle de comb[1] (1)
// puis celle de comb[2] (2)

// APRES LA DEUXIEME EXECUTION DE ft_recursive_combn
// comb[0] vaudra 0
// comb[1] vaudra 1
// comb[2] vaudra 2

// puis ft_write_combn écrira la valeur de comb[0] (0)
// puis celle de comb[1] (1)
// puis celle de comb[2] (2)

// etc.
// (voir plus bas)

// ATTENTION A INDIQUER *comb
// correspondant à l'adresse du tableau comb
// (plus précisément, l'adresse de son premier élément comb[0])

// n correspondra à la valeur n passée lors de l'appel

// pos sera initialisé ici
// et prendra la valeur 0 (passée en 3ème paramètre)

// prev sera initialisé ici
// et prendra la valeur -1 (passée en 4ème paramètre)

// ATTENTION :
// CETTE FONCTION EST RECURSIVE !
void ft_recursive_combn(int *comb, int n, int pos, int prev)
{
    // on déclare localement un entier i
    // i SERA DECLAREE A CHAQUE APPEL DE CETTE FONCTION !
    // une nouvelle instance locale de la variable i
    // sera en fait créée dans la pile d'appels de la fonction !

```

```

// donc chaque appel récursif de la fonction
// travaillera avec SA PROPRE COPIE DE LA VARIABLE i !

// cette variable sera utilisée comme un compteur
// pour itérer à travers les chiffres pouvant être ajoutés
// à la combinaison actuelle
int    i;

// IMPORTANT :
// CONDITION DE SORTIE DE LA FONCTION RECURSIVE !
// A INDIQUER EN PREMIER DANS LA FONCTION POUR PLUS DE LISIBILITE
// ET POUVOIR STOPPER L'EXECUTION DU PROGRAMME DES QUE NECESSAIRE

// si tous les chiffres de la combinaison ont été générés
// (par exemple, 0, 1 et 2 pour n = 3
// (pour la première combinaison))

// AUTREMENT DIT, SI UNE COMBINAISON A ETE GENEREE !
// par exemple 012
if (pos == n)
{
    // on écrit la combinaison (voir plus bas)
    ft_write_combn(comb, n);

    // POUR ECRIRE ", " ENTRE CHAQUE COMBINAISON SAUF LA DERNIERE

    // SI LA DERNIERE COMBINAISON N'EST PAS ENCORE ATTEINTE
    // (si le premier chiffre de la combinaison générée
    // est inférieur à 10 - n
    // pour n = 3 : 10 - n = 10 - 3 = 7)
    // donc si le premier chiffre de la combinaison n'est pas égal
    // à 7)
    // CAR LA DERNIERE COMBINAISON POSSIBLE POUR n = 3 EST 789
    // ET L'AVANT-DERNIERE EST 689
    if (comb[0] < 10 - n)

        // on écrit ", "
        write(1, ", ", 2);
}

```

```

        // on termine l'exécution de la fonction ft_recursive_combn
        return ;
    }

    // DANS TOUS LES CAS (SAUF SI LA DERNIERE COMBINAISON A ETE ATTEINTE)

    // on définit i comme étant égal à prev + 1
    // prev est initialement égal à -1
    // donc i sera initialement égal à 0
    i = prev + 1;

    // tant que i est inférieur à 10
    // (pour que i parcoure les chiffres de 1 à 9)
    while (i < 10)
    {
        // pos est initialement égal à 0
        // donc comb[pos] correspondra à comb[0]
        // donc le premier chiffre de la combinaison
        // i est initialement égal à 0
        // donc lors de la première exécution
        // comb[0] = 0
        comb[pos] = i;

        // on exécute à nouveau la fonction ft_recursive_combn
        // avec pos = pos + 1
        // et prev = i
        // (donc avec pos = 1
        // et prev = 0)
        // (après avoir défini le premier chiffre de la combinaison à 0
        // on passe au deuxième chiffre de la combinaison)
        // i = prev + 1 => i = 1
        // comb[1] = 1
        // PUIS
        // pos = 2
        // prev = 1
        // i = 2
        // comb[2] = 2
        // PUIS
    }

```

```

        // pos = 3
        // prev = 2

        // ATTENTION : pos == n
        // on écrit la combinaison puis on arrête l'exécution de la fonction récursive
        // on revient donc à la fonction récursive du niveau précédent
        // (dont les valeurs sont :
        // pos = 2
        // prev = 1
        // i = 2
        // comb[2] = 2)
        // pour la continuer
        // à la ligne i++
        // donc i = 3
        // comb[pos] = i => comb[2] = 3
        // on exécute ft_recursive_combn
        // pos == n donc on écrit 013
        // on revient au niveau précédent
        // etc
        // jusqu'à ce que i = 10
        // on revient donc au niveau précédent
        // avec pos = 1
        // prev = 0
        // i = 1
        // i++
        // donc i = 2
        // etc...
        ft_recursive_combn(comb, n, pos + 1, i);
        i++;
    }
}

// fonction pour écrire les combinaisons en parcourant le tableau comb
// on écrira comb[0], comb[1], comb[2]
void ft_write_combn(int *comb, int n)
{
    int    i;

```

```
        i = 0;
        while (i < n)
        {
            ft_write_number(comb[i]);
            i++;
        }
    }

// voir ex05
void    ft_write_number(int n)
{
    char    c;

    c = n + '0';
    write(1, &c, 1);
}
```