```
// cette fonction retourne un pointeur vers
// l'adresse du premier élément du tableau d'entier,
// qui contient toutes les valeurs entre min et max
// (min inclus ; max exclus)
// par exemple, pour min = 5 et max = 10
// le tableau d'int retourné contiendra les éléments :
// 5, 6, 7, 8 et 9
// ce tableau sera créé dynamiquement

// ATTENTION :
// si la valeur de min est supérieure ou égale à celle de max,
// alors un pointeur nul (NULL) sera retourné

// pour pouvoir utiliser malloc
#include <stdlib.h>

// la fonction prend en paramètre l'entier min,
// l'entier max, et retourne un pointeur
// vers le tableau d'entiers créé dynamiquement
int     *ft_range(int min, int max)
{
        // taille (en nombre d'éléments) du tableau d'entiers
        // à créer
        int     length;

        // le tableau d'entiers à créer et à retourner
        int     *tab;

        // indice pour parcourir le tableau d'entiers
        // et y insérer au fur et à mesure les entiers
        // de min à max
        int     i;

        // si min est supérieur ou égal à max
        if (min >= max)
                // on retourne un pointeur nul
                return (NULL);
```

```
// nombre d'entiers à insérer dans le tableau
// EXEMPLE avec min = 5 et max = 10
// 10 - 5 = 5
// le tableau devra contenir :
// 5, 6, 7, 8, 9
// on a bien 5 éléments
length = max - min;

// on alloue, en octet, length fois la taille d'un élément pointé par tab
// à la variable tab, tableau d'int
// donc, pour length = 5, 5 fois la taille d'un entier
// (en général, un entier est codé sur 4 octets)
// pour cet exemple, on allouera alors 5 * 4 = 20 octets à tab
// on convertit explicitement le pointeur en un int *
// (tableau d'entiers, plus précisément pointeur sur un entier
// (sur le premier élément d'un tableau d'entiers))
tab = (int *)malloc(length * sizeof(*tab));

// si l'allocation a réussi
// (si 20 octets étaient disponibles en mémoire)
if (tab)
{
        // on initialise i à 0
        // (pour commencer à l'indice 0 du tableau)
        i = 0;

        // tant que min est inférieur à max
        // (car on veut min inclus mais max exclus)
        // on s'arrête avant max
        // on déclare la valeur min
        // comme premier élément du tableau
        // (5 dans notre exemple)
        // on incrémente min de 1
        // (pour passer à l'entier suivant directement min
        // 6 dans notre exemple
        // puis on avance à l'emplacement du prochain indice dans le tableau
        // on ajoute le nouveau min (6 dans notre exemple) en deuxième
        // position dans le tableau, etc
```

```
                // jusqu'à max - 1 compris
                // (9 dans notre exemple)
                while (min < max)
                {
                        tab[i] = min;
                        min++;
                        i++;
                }
                // on retourne le tableau
                return (tab);
        }
        // on retourne NULL (pointeur NULL)
        // si l'allocation n'a pas réussi
        return (NULL);
}

#include "ft_range.h"
#include "ft_putnbr.h"
#include <unistd.h>
#include <stdlib.h>

int     main(void)
{
        int     min;
        int     max;
        int     *tab;
        int     i;
        int     length;

        min = 12;
        max = 42;
        tab = ft_range(min, max);
        if (tab == NULL)
                return (1);
        length = max - min;
        i = 0;
        while (i < length)
        {
```

```
                ft_putnbr(tab[i]);
                write(1, "\n", 1);
                i++;
        }
        free(tab);
        tab = NULL;
        return (0);
}
```