```
// cette fonction convertit la chaine de caractères str,
// qui est dans la base indiquée par base,
// en entier (int)
// cette fonction reproduit le comportement de ft_atoi (voir ex03)
// la partie pour la gestion des erreurs est la même que
// pour ft_putnbr_base (ex04)

#include <unistd.h>

// pour calculer le nombre de caractères
// dans la chaine de caractères base
// voir ex04
int     ft_base_len(char *base);

// fonction de gestion des erreurs
// voir ex04
int     ft_check_base_error(char *base, int base_len);

// ft_atoi_str_base_to_dec prend en paramètre la chaine de caractères
// pointée par str
// qu'elle doit convertir
// elle retourne un int (qui correspond à sa conversion)
// voir ex03
// ATTENTION !
// contrairement à ft_atoi, qui extrait le nombre
// trouvé dans str en entier en base 10
// ft_atoi_str_base_to_dec convertit la chaine de caractères
// trouvée dans str en entier, qui est dans la base indiquée par base,
// en entier en base 10
// (en passant chaque caractère extrait à ft_char_base_to_dec)

int     ft_atoi_str_base_to_dec(char *str, int base_len, char *base);

// fonction pour convertir un caractère dans la base donnée
int     ft_char_base_to_dec(char c, int base_len, char *base);

// voir ex04
// (sans la partie gestion des nombres négatifs)
```

```
int     ft_atoi_base(char *str, char *base)
{
        int     base_length;
        int     error;
        int     nbr;

        base_length = ft_base_len(base);
        error = ft_check_base_error(base, base_length);
        if (error)
                return (0);
        nbr = ft_atoi_str_base_to_dec(str, base_length, base);
        return (nbr);
}

// voir ex04
int     ft_base_len(char *base)
{
        char    *ptr;

        ptr = base;
        while (*ptr != '\0')
                ptr++;
        return (ptr - base);
}

// voir ex04
int     ft_check_base_error(char *base, long int base_len)
{
        char    *ptr1;
        char    *ptr2;

        if (base_len == 0 || base_len == 1)
                return (1);
        ptr1 = base;
        while (ptr1 < base + base_len - 1)
        {
                ptr2 = ptr1 + 1;
                while (ptr2 < base + base_len)
```

```
                {
                        if (*ptr1 == *ptr2)
                                return (1);
                        ptr2++;
                }
                ptr1++;
        }
        ptr1 = base;
        while (ptr1 < base + base_len)
        {
                if (*ptr1 == '+' || *ptr1 == '-')
                        return (1);
                ptr1++;
        }
        return (0);
}

// voir ex03
// REMARQUE :
// la gestion du signe a été optimisée
// par rapport à ex03 :)
int     ft_atoi_str_base_to_dec(char *str, int base_len, char *base)
{
        int     sign;
        int     n;
        int     nbr;

        sign = 1;
        nbr = 0;
        while ((*str >= 9 && *str <= 13) || *str == 32)
                str++;
        while (*str == '-' || *str == '+')
        {
                // on inverse le signe à chaque fois
                // qu'un signe '-' est recontré
                if (*str == '-')
                        sign = -sign;
                str++;
```

```
}

// ATTENTION :
// on extrait tous les caractères suivants
// jusqu'à la fin de la chaine
// (et non pas uniquement les chiffres (caractères de 0 à 9))
while (*str != '\0')
{
        // on convertit le caractère en cours
        // (le premier caractère formant le nombre
        // dans la base donnée lors du premier passage dans
        // la boucle, etc)
        // en sa valeur décimale
        // voir ft_char_base_to_dec
        n = ft_char_base_to_dec(*str, base_len, base);
        // on décale le premier chiffre n trouvé
        // (dans la base correspondante)
        // vers la gauche pour y ajouter le chiffre n suivant
        // voir ex04
        // en base 10, pour déplacer un chiffre d'une position
        // vers la gauche, il faut multiplier ce chiffre par 10
        // ex : 2 * 10 devient 20
        // on peut donc ajouter le n suivant en l'additionnant
        // à cette valeur
        // MAIS en base 16 par exemple
        // pour déplacer un chiffre d'une position
        // vers la gauche, il faut multiplier ce chiffre par 16
        // ex : 2 * 16 devient 32
        // on multiplie donc nbr par base_len
        // et non pas par 10

        // ATTENTION :
        // il faut gérer le cas où le caractère en cours
        // n'a pas été trouvé dans la base
        // (si le caractère n'est pas valide dans la base)
        // par ft_char_base_to_dec(*str, base_len, base);
        // ici aussi !
        // dans ce cas, on doit retourner une erreur
```

```
                // car le nombre n'est pas convertible en base 10 correctement !
                // on retourne donc 0
                // (comme pour ft_check_base_error)
                if (n == -1)
                        return (0);

                // ATTENTION :
                // il ne faut pas décaler le premier chiffre à gauche !
                // mais comme nbr vaut 0 au départ
                // on a nbr = 0 * base_len + n
                // donc nbr = n
                // c'est donc correct

                nbr = nbr * base_len + n;

                // on passe au caractère suivant, pour sa conversion
                // puis son addition à nbr
                // pour former au fur et à mesure le nombre en base 10
                str++;
        }
        // on ajoute le signe à nbr
        // puis on retourne le tout
        return (sign * nbr);
}

// c est le caractère en cours
// base_len la longueur de la base
// base la chaine de caractères représentant la base
// (0123456789ABCDEF par exemple)
int     ft_char_base_to_dec(char c, int base_len, char *base)
{
        // ptr stocke l'adresse du début de la base
        // pour parcourir base
        // et ainsi garder base à l'adresse du premier
        // caractère de la chaine
        char    *ptr;

        ptr = base;
```

```
        // tant que l'on a pas dépassé l'adresse du dernier caractère
        // de la chaine de caractères base
        // (cette boucle permet de parcourir base du premier
        // au dernier caractère)
        while (ptr < base + base_len)
        {
                // si le caractère en cours est identique
                // à celui pointé par ptr
                // (donc si il y a correspondance entre les 2 caractères,
                // s'il a été trouvé dans la base)
                if (c == *ptr)
                        // on retourne l'index de c dans la base
                        // (en retournant la différence entre l'adresse
                        // du caractère de base trouvé
                        // et celle du début de base)
                        // exemple :
                        // c = A
                        // base = 0123456789ABCDEF
                        // c == *ptr => ptr = adresse du 11ème caractère de base
                        // (= index 11)
                        // base = adresse du 1er caractère de base
                        // (= index 1)
                        // index 11 - index 1 = 10
                        // on retourne 10
                        // A correspond bien à 10 en base 10 !
                        return (ptr - base);
                // on passe au caractère suivant de base
                // s'il ne correspond pas au caractère en cours
                ptr++;
        }
        // on retourne -1 si la base a été parcourue sans avoir trouvé
        // le caractère c
        return (-1);
}

#include "ft_strcpy.h"
#include "ft_atoi_base.h"
```

```c
#include "ft_putnbr.h"

// RESULTAT : 342391
int     main(void)
{
        char    str[9];
        char    base[9];
        int             nbr;

        ft_strcpy(str, "poneyvif");
        ft_strcpy(base, "poneyvif");
        nbr = ft_atoi_base(str, base);
        ft_putnbr(nbr);
        return (0);
}
```