Example:
char *nbr = "207a cEf";
char *base_from = "0123456789"
int i = 0;

nbr[0] = '2';
compare the value of nbr[0] VS base_from[i] (range of i 0 to 9)
nbr[0] = base_from[2];
so i = 2 return i immediately
samething for nbr[1] = 0 nbr[2] = 7
nbr[3] = 'a' no base_from[i] value equals nbr[3] return (-1) at the end of function find.

int find(nbr_from, base_from)

Example of correct base format:
base_from / base_to = "01" binary_base
base_from / base_to = "01234567" octal_base
base_from / base_to = "0123456789" decimal_base
base_from / base_to = "0123456789ABCDEF" hexadecimal_base

Example of incorrect base format:
base_from / base_to = "011" same number in the string base
base_from / base_to = "1" or "" only one number or empty in the string base
base_from / base_to = "+- 01" "+" or "-" or " " included int the string base

char *base_from

char *base_to

int ft_atoi_base(char *nbr_from, char *base_from, int len_from)

int check_base(char *base)

Example:
char *nbr = "207a cEf";
char *base_from = "0123456789"
int len_from = 10; (return value from the int check_base(base_from)
int num_decimal = 0;
num_decimal = num_decimal * len_from + find(nbr_from, base_from) % len_from

num_decimal = num_decimal * 10 + find(nbr_from[0], base_from) % 10 = 0 * 10 + 2 % 10 = 2;
num_decimal = 2 * 10 + 0 % 10 = 20;
num_decimal = 20 * 10 + 7 % 10 = 207;
nbr[3] = 'a' so the return value of find(nbr_from[3], base_from) is -1
if we meet a negative return value like "-1" we return the value of num_decimal immediately!
the return value of this example is num_decimal = 207

if base format is correct, the function will return the real length of base, else it will return the false length of base (-1)

2

2

check_base (base_to)

check_base (base_from)

base_format correct return value length of base_to

base_format incorrect return value -1

base_format incorrect return value -1

base_format correct return value length of base_from

Example:
nbr = "- -+ - 234";
nbr = "11001101".
nbr = "0278";
nbr = " --207a cEf";

char *nbr

Example:
nbr = "234"; sign = - 1
nbr = "11001101". sign = 1
nbr = "0278"; sign = 1;
nbr = "207a cEf"; sign = 1;

char *nbr (without "+" "-" " ")

3

int num_decimal

char *ft_convert_base(char *nbr, char *base_from, char *base_to)

5

char *num_to

void ft_put_nbr(char *base_to, char *num_to, int num_dec, int sign)

Example:
char *nbr = "- - - +207a cEf";
char *base_to = "01"
char *num_to (without value inside but have 9 spaces to put characters)
int num_decimal = 207
int sign = -1 (because we have 3 "-" in nbr, it is an even number, so sign = -1)
check_base(base_to) = 2 => length of base_to
num_to[memory_size] = num_to[9] = '\0'
num_to[8] = num_decimal % length of base_to = 207 % 2 = 1 ; num_decimal = num_decimal / length of base = 207 / 2 = 103
num_to[7] = 103 % 2 = 1; num_decimal = 103 / 2 = 51
num_to[6] = .51 % 2 = 1; num_decimal = 51 / 2 = 25
num_to[5] = 25 % 2 = 1; num_decimal = 25 / 2 = 12
num_to[4] = 12 % 2 = 0; num_decimal = 12 / 2 = 6
num_to[3] = 6 % 2 = 0; num_decimal = 6 / 2 = 3
num_to[2] = 3 % 2 = 1; num_decimal = 3 / 2 = 1
num_to[1] = 1 % 2 = 1; num_decimal = 1 / 2 = 0
since sign = -1, so num[0] = '-'
 we transferred the *num_to by argument char *num_to to the function ft_convert_base

int sign
sign = 1;

char *ft_sign_check_nbr(char *nbr, int *sign)

odd

sign = -1

even

sign = 1

Purposal :
Skip all the "+" "-" " " before the next step.
Determine the quantity of "-" before starting the true number (odd or even?)

1

check_base(base_to) or check_base(base_from) = = -1 then return (Null)
after ft_sign_check_nbr(char *nbr, int *sign),
we use ft_atoi_base, if *nbr == 0 (means that after skip "-" "+" " ", the 1st character of nbr is not compatible with base_from so the ft_atoi_base will return 0 as the value of nbr. in this case return (Null) (ps : char *nbr = "0" not included in this case)
after using malloc for num_to, if num_to == Null, return (Null)

int memory_size

memory_size = 9

4

int measure_memory(char *base_to, int num_dec, int sign)

Example:
char *nbr = "- - - +207a cEf";
int num_decimal = 207
int sign = -1 (because we have 3 "-" in nbr, it is an even number, so sign = -1)
char *base_to = "01"
icheck_base(base_to) = 2 => length of base_to
Int memory_size = 0
207 / 2 = 103 memory_size = 1
103 / 2 = 51 memory_size = 2
51 / 2 = 25 memory_size = 3
25 / 2 = 12 memory_size = 4
12 / 2 = 6 memory_size = 5
6 / 2 = 3 memory_size = 6
3 /2 = 1 memory_size = 7
1 / 2 = 0 memory_size = 8
we need divided by 2 by 8 times.
so the result is 8.
since sign = -1
so we add 1 for memory_size.
the return value is 9 (memory_size = 9)