Homework #2: my_malloc and my_free (100)

Submit *only* **my_malloc.c**

Your job is to write a *simplified* version of **malloc( )** and **free( )** in **C**. Specifically, you must implement **my_malloc(…)** and **my_free(…)** using the **provided** header file (i.e., *my_malloc.h*). In particular, you must utilize the provided **struct** (i.e., free_list_node) and implement three functions:

```
void *my_malloc( int size );
void *my_free( void *ptr );
void print_free_list();
```

Your implementation of *my_malloc* **must:**

- print each step to ***stderr*** (e.g., scanning free list, calling sbrk( ), etc.)
- use the "**first fit**" paradigm
- guard against negative sizes
- guard against sizes greater than **`SIZE`** (in header)
- return a pointer to new memory (return NULL on error)
- **NOT** call the built-in malloc( ) function

Your implementation of *my_free* **must:**

- print each step to ***stderr***
- append the deallocated space to the **end** of the free list
- **NOT** call the built-in free( ) function

print_free_list( ) function must show details (i.e., address, size, address of next node) of the current free list.

NOTE: I will test your code (*my_malloc.c*) with the ORIGINAL header file (*my_malloc.h*) and my own *main.c* => In other words, **DO NOT** modify the provided *my_malloc.h* file or submit code with a *main( )* function. Your code must compile and run using the **gcc** compiler (not g++).

HINTS:

- `sbrk( … )`
- `sizeof( free_list_node )`
- `int *x = (int *) my_malloc(10 * sizeof(int));`
- pointer magic and casting
- draw lots of pictures
- plan, plan, plan before you code
- think of edge cases
- write your own `main.c` to test (but do not submit it)
- `0x%x` placeholder for `fprintf( … )`

EXAMPLE (NOTE: sbrk(…) called with size **2048**)

=> after allocating an array of 100 doubles, why am I a left with **1216** bytes on the free list?

```
UNIX> ./a.out
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
===========================================================
main: allocating array of 100 doubles
my_malloc: called with size = 800
my_malloc: allocating new free list
my_malloc: scanning free list...found space in free list.
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
       0 | 0x e5f1300 |     1216 | 0x       0
===========================================================
main: allocating array of 100 integers
my_malloc: called with size = 400
my_malloc: scanning free list...found space in free list.
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
       0 | 0x e5f2d00 |      800 | 0x       0
===========================================================
main: allocating array of 1000 chars
my_malloc: called with size = 1000
my_malloc: scanning free list...no space in free list.
my_malloc: calling sbrk() to expand heap...
my_malloc: scanning free list...found space in free list.
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
       0 | 0x e5f2d00 |      800 | 0x e5f2780
       1 | 0x e5f2780 |     1016 | 0x       0
===========================================================
main: freeing chars
my_free: called with 0x e5ee900, size =     1000
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
       0 | 0x e5f2d00 |      800 | 0x e5f2780
       1 | 0x e5f2780 |     1016 | 0x e5ee800
       2 | 0x e5ee800 |     1000 | 0x       0
===========================================================
main: freeing ints
my_free: called with 0x e5f1400, size =      400
main: printing free list:
================FREE LIST==================================
  NODE # |    ADDRESS |     SIZE |       NEXT
       0 | 0x e5f2d00 |      800 | 0x e5f2780
       1 | 0x e5f2780 |     1016 | 0x e5ee800
       2 | 0x e5ee800 |     1000 | 0x e5f1300
       3 | 0x e5f1300 |      400 | 0x       0
===========================================================
main: freeing doubles
```

```
my_free: called with 0x e5ee100, size =      800
main: printing free list:
================FREE LIST=================================
  NODE # |    ADDRESS |      SIZE |       NEXT
       0 | 0x e5f2d00 |       800 | 0x e5f2780
       1 | 0x e5f2780 |      1016 | 0x e5ee800
       2 | 0x e5ee800 |      1000 | 0x e5f1300
       3 | 0x e5f1300 |       400 | 0x e5ee000
       4 | 0x e5ee000 |       800 | 0x        0
==========================================================
UNIX>
```