MTL 458, Operating Systems, Homework 2

In this assignment, you are asked to design your own version of memory allocator including `malloc()` and `free()` functions in C, which can be used as replacement to the original functions. It will require implementing a lot of concepts taught in the class. The details are given below. The assignment will be worth 10 points, like the last one.

Requirements: You need to construct following five functions.

a) `my_init()`: This will be the initialization function for the memory allocator. It will use the `mmap` system call to request a page of 4KB from the memory and initialize any other data structure as required. The 4KB page is the space that the memory allocator will be working with. The user will first invoke `my_init()` before requesting any memory. It should return 0 on successful completion, otherwise the error code. This would require you to understand and use the `mmap` and `munmap` system calls, which is also part of the assignment. [1.5 points]

b) `my_alloc(int)`: This will be your version of `malloc(int)`. Any space asked for should be a multiple of 8 bytes, otherwise the function returns NULL, which is also does in case of any other error. As expected, just like `malloc(int)`, it would return a pointer to the address assigned in case of successful completion. [3.5 points]

c) `my_free(char *)`: Takes a pointer to a previously allocated chunk and frees up that chunk, just like `free(char *)` does. [2.5 points]

d) `my_clean()`: This would be done at the end to return the memory back to the OS using `munmap` system call. [0.5 point]

e) `my_heapinfo()`: This would give some useful information about the current state of heap. This must include the maximum size allowed for the heap, the size of the current heap, total size of the free memory in the heap, number of blocks allocated, size of a smallest available chunk, and the size of a largest available chunk. Observe that to implement this you might have to traverse the whole free-list. Bonus marks if you can maintain some data structure which does not have to do that every single time (even though sometimes it will have to, if it is a linked-list). [2 point]

Guidelines: While doing the assignment, please be careful about the following.

- You will submit a `.zip` file `<Entry_No>_A2.zip` which would contain the file `my_alloc.c`.
- Your code must **not** use the `malloc` or `free` functions or their family (`realloc` etc.). You are supposed to implement them writing your own code.
- As we had discussed in the class, the free-list will be embedded inside this 4KB of memory.
- `my_alloc(int)` and `my_free(char *)` should perform as desired (they should fo splitting and coalescing).
- You can use any of the strategies mentioned in the class (best-fit, worst fit etc), but do mention them as comments when you write your `my_alloc(int)` function.