(MeMS) Memory Management System

Initialize MeMS (mems_init()):
This function sets up the memory system.
It creates the initial structure to manage memory(`head/shed)`.
mhead: main label for our storage area.
shead: It's a sub-label within our storage, pointing to smaller sections.

Finish Using MeMS (mems_finish()):
This function is like cleaning/deleting up after using the memory(by using mnump) .
It frees the memory and prepares the system for closure.
It's similar to tidying up a room before leaving.
mainList *node: tracking each section we used for storage.

Allocate Memory (mems_malloc(size_t size)):
This function gets memory and It finds available space create manelist and sublist. And by that if we are running out of space so it will crate for us.
mDummy, sDummy: to look for available space in our storage area.
virtualCounter: It's a counter that helps keep track of how much space we've used.

Check MeMS Stats (mems_print_stats()):
This function tells you about the memory used. It gives information about how much memory is used and how much is free.
dummyHead: to check each storage section and see how much is full or empty.

Get Physical Address (mems_get(void *v_ptr)):
the actual location of stored nodes
It's like finding the exact spot where you stored something in your storage unit.

Free Memory (mems_free(void *v_ptr)):
This function returns the memory you don't need anymore.
It tells the system that you're done using a particular memory space, and it can be used by someone else. It's like giving back the node that was borrowed.
dummyHead:mark the shelves as free for others to use.

//test cases
As Given in the test file(ScreenShot or outPut) when we allocate memory through the mems_melloc it provide a virtual address to the user and give Memory to the variable according to the diamond of meme_melloc, it also keep track of how much size it has given and where it is in actually memory (os given Virtual address) all this we store in the linked list data structure.

This custom malloc will work completely for some basic cases like providing memory to different processes and taking care less fragmentation.
It detect continuous hole and join them(eg h1[300:399]<-->h2[400:499] = h1[300:499],and make the larger hole (so that a large hole process can come and set there) and it also split the lager node to the hole and process according to the size requirement eg (mems_melloc diamonds 50 bytes but we have 100 bytes then mems_melloc will divide 100 bytes in 50 bytes of block which of each and will make one hole and one process.

Command to run:
For compiling : make
For running executable : ./example