Readme

Structure Info:
Two structures main_node and sub are created for maintaining the free list. The memory for these structures is allocated from a set of pages pointed to by the curr pointer, which is a global pointer. Rem_mem_struct is another global variable that counts how much memory is left on the current page. If the remaining memory is less than the memory required to create a structure, a new page is allocated using mmap and the curr pointer is updated accordingly.

The main node contains the pointer to the starting address of the page used to allocate memory for user processes and also the starting and ending virtual address of the page.
The sub-structure stores the starting and ending physical and virtual addresses of the memory block it is pointing to and its size.

Logical behind the functions:

Mems_alloc():
1. at the beginning, the head pointer of mainNode points to NULL, so calculating the size demanded in the multiple of page size and storing it in the page_ptr of the head and demanding the space calculated using the func of size for storing the free list structure
2. if the head is not NULL, find if the appropriate size of the hole is present or not if present, then partition the space into two parts, one which will become the process demanded by the user and the other is the hole if there is no hole, then creating the pageSize and new mainNode if the size for the free list is full then allocating more space and updating the details required and storing it.

Mems_finish():
Traversing through the list using head-> next pointer to free the memory using munmap and doing the required error handling of the munmap

mems_get():
Traversing through the list and finding the appropriate position using vir_start,vir_end,mems_vir_start, and mems_vir_end and returning the physical address calucaluted by this line -> “checker - chain->mem_vir_start + chain -> mem_phys_start”

mems_free():
Traversing through the list and finding the correct position using the above logic and marking the process as a hole and then again traversing through the mainNode list to find if there are consecutive holes present or not in the substructure and if there are, then merging it as one hole.

Mems_print_stats(): printing the required details to inform the user what is going on in the backend when they use the mems function to demand some space or remove the space created by it.