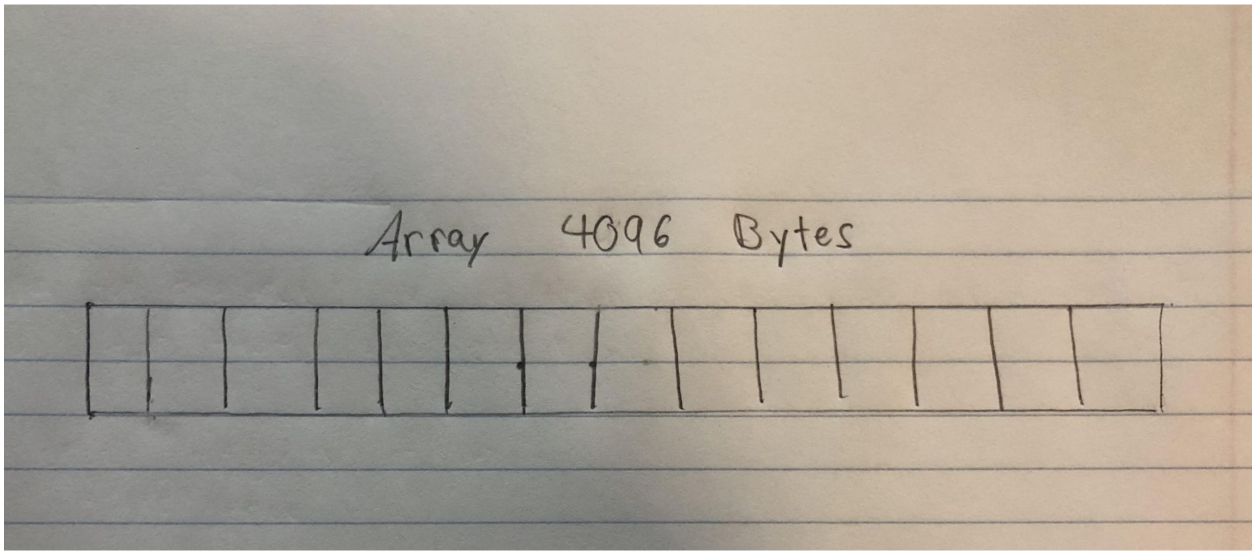
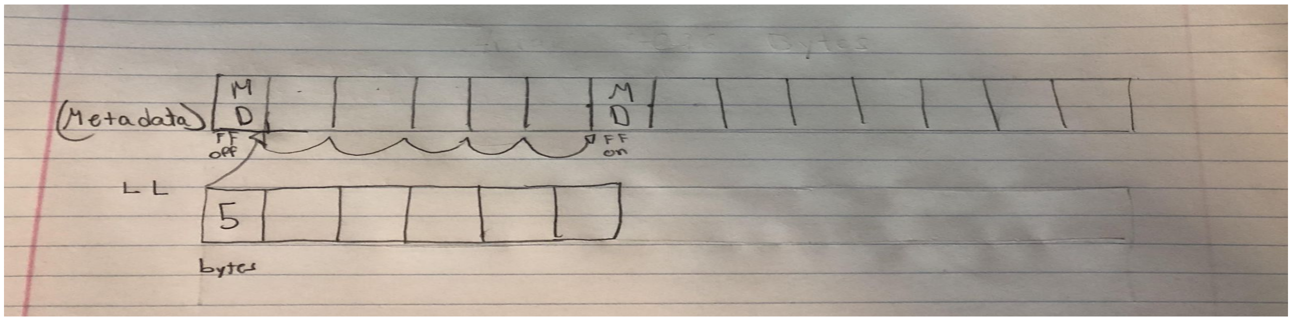
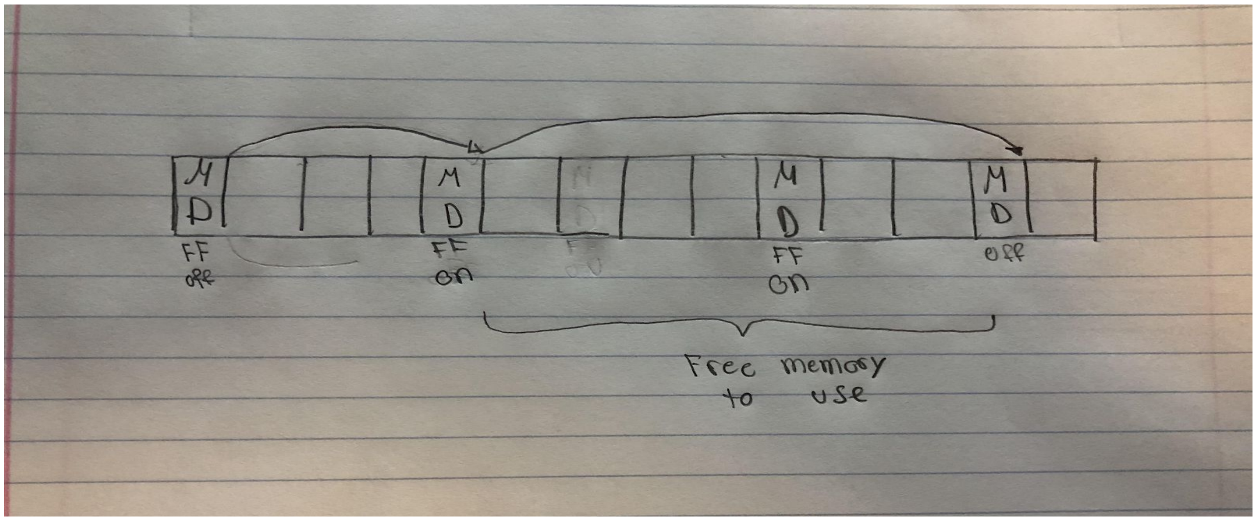
The way we created our malloc algorithm is we simulated a heap in the form of an array. The array holds 4096 bytes which represents the total amount of memory that we have in order to malloc whatever the user needs. The way we malloc our memory is by having a pointer that is represented as a metadata bookmark. What we do is, we initialize the start of the metadata link list which is inserted into the beginning of the array of myblock, this is where the start of the first malloc will be called.



When the user calls to malloc a set of data, the malloc function will traverse the metadata list and search for two consecutive metadata structs with their free flag on. If there is no current flag, to the point where the Linked List finishes traversing the array, then it will add the metadata bookmark to indicate the end of malloc.

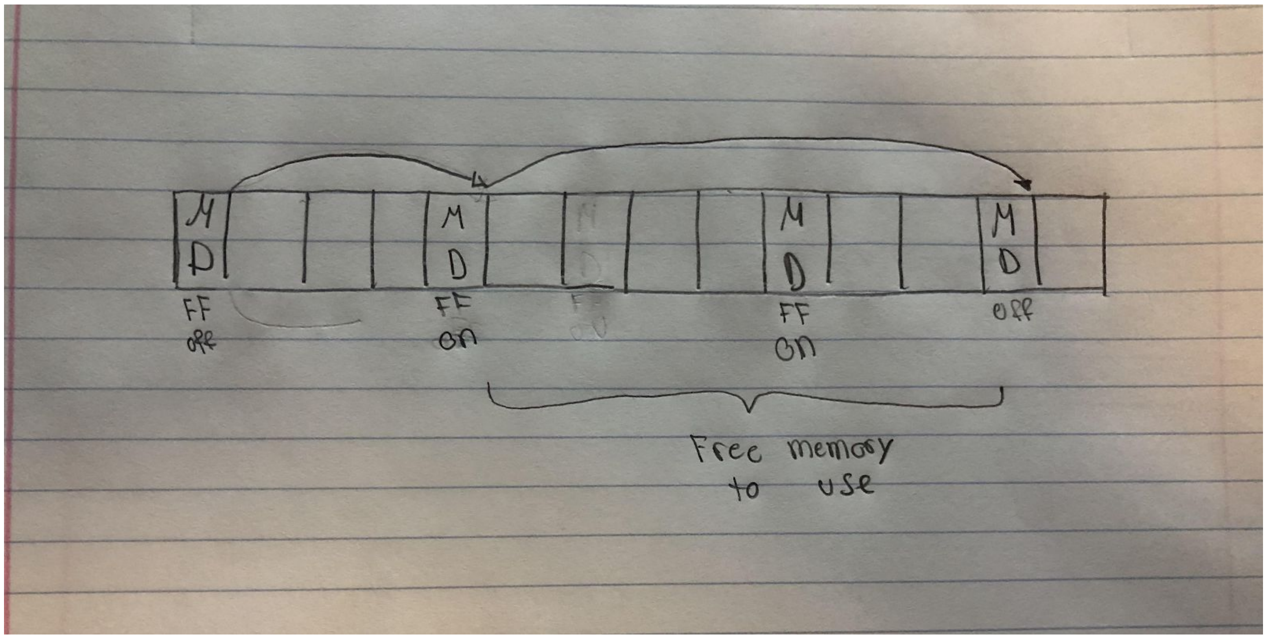


In this case, we call our combineBlocks function which combines these two blocks for the best possible chance to fulfill the malloc request. This is done by adding up the sizes of the two blocks and setting the initial metadata pointer to point to what is ahead of the second metadata struct. The malloc function then traverses the metadata list again in search of first metadata that is free and has an adequate size to fulfill the request. If it is exactly the size required, we simply set the metadata free flag to zero and return the address to the position ahead of the metadata.



Otherwise we call our insertMetadata function. What this will do is insert a Metadata struct, bookmarking the end of the requested data while also indicating the remainder space of the free block ahead. If no adequate space can be found we simply return an error and return NULL or if there was no more amount of memory, then we would return "Insufficient memory to fulfill malloc request…".

What our Free Function will allow the user to free up the memory that they have previously called. It will traverse the array for that specific memory address and once it finds it, it will set the metadata free flag to on, indicating that that memory is free to use. As previously stated, if there are two consecutive free flags that are on, then the combine blocks function will check for that. If it is true, then the previous metadata free flag that is off, will point to the next free flag that is on, indicating that all the memory that is in between the two metadata, will be available for use, when the next malloc() is called.



.