READ ME

My implementation of malloc() uses the first fit algorithm. I create a struct called memBlock which holds two pointer, next and prev to point to successor blocks and predecessor blocks. The struct also contains a size field to indicate the size of the allocated block of memory as well as a isFree field to indicate whether the block Is free or not. I create a static array of size 5000 bytes which I use as my memory heap. I cast the array as type struct in order to place my memBlock nodes within it. From this point I test for whether the size requested by the user is larger then my heap, whether the block we are on is free or not, whether we are at the end of the list and we can return the memory but we cannot include the header, and lastly I simply insert the memory in the location that it fits.

For the free method I simply loop through the nodes and compare the pointer given to free to the memBlock pointer in the array. If there is a match then I free the node otherwise we have an error where perhaps the pointer was already free'd and or never allocated and or we where given a location out of bounds of the array or it was within the middle of a block of memory.

I also have created a helper function to allow the merging of blocks. In my free function when I free blocks I call my mergeFreeBlocks function to check to the left and right for free blocks and then merge them together. I then recurse and keep checking until I no longer find free blocks.

For malloc whenever a block of memory is allocated I output that it was allocated with the size appended and for free output that a block was freed and I append the block offset.

Hopefully everything works I really need the points. I DID NOT IMPLEMENT ANY OF THE EXTRA CREDIT.