MyMalloc and MyFree writeup:

Coding by Liam O'Brien & Taner Doganay

Makefile: Our makefile simply automates to process of compiling the objects and includes a script that automatically cleans up directory of objects for testing.

Mymalloc.h: This file replaces any instances of malloc and free with our own version of malloc and free called mymalloc and myfree. It also declares our node for which we organize our nodes. This file also declares our memorysize for each node.

Mymalloc.c: This file firstly creates an index to find the first free index that does not contain any entry. Next, we have our myMalloc function which is used to allocate memory for the user. The first check is to make sure that the user does not try to allocate o bytes of memory. The next part is the function to initiate myMalloc for the first time. We then set the pointer to the first free index of our 5,000 byte index. If myMalloc has already been called, we start a loop first to check if the size the uer is trying to malloc is too small or if the pointer is not free. Next, if the chunk being malloced is the accurate size but there is not enough memory to hold a new head node, we return a pointer the size of the node. Lastly if all else fails, we malloc enough space for the data and the node. If this loop fails, it means that there is not enough free memory to be allocated, so we print out an error to the user stating so.

The next method is our "myfree" First, we check to see if the pointer is pointing to null and if so we print an error stating so, because there is no reason to try and free it. Next, we check for a valid memory pointer. In case there is a pointer that was not malloced, we prompt the user. We then proceed to check that the current pointer is not already freed. The last step is to merge all of the remaining nodes together so we do not have random nodes of data in memory. It first tries to merge with the previous nodes in the Node, then in looks forward to see if there are any following free nodes that it can merge with.