Jake Karasik

Nicholas Petriello

mymalloc.h

This contains the rudimentary tools for the mymalloc.c functions.  It contains the struct of metadata_ which we use to delineate certain blocks of information.  It has a is_set variable that shows whether it is free or not.  It has a size that determines the amount of space allocated.  The id is a flag that ensures that the ptr being sent to free is a valid pointer.  And adhering to the linked list architecture there is a next ptr to the subsequent nodes that set the boundaries of allocated memory in our simulated memory.

We also have prototypes for mymalloc( ), myfree( ), resetmyblock( ).

mymalloc.c

This contains the functions and included libraries.  We have global variables myblock representing the memory space and remaining_space representing the space left to allocate.

We have 3 functions:

resetmyblock sets the memory back to its original empty space so that we may run the tests without overlapping of workloads affecting program times..

mymalloc takes in a size and returns a pointer to allocated memory space.  If the size is zero, a null pointer is returned.  It creates a metadata ptr that traverses a linked list delineating memory blocks and finds an available block.  If a list doesn't exist, a head is made and starts it. It also sets the variables for each node in the list.

myfree just sets the is_set metadata for a node so that it is deemed available space by mymalloc.  If a null value is given an error is printed.  myfree finds the index of memory and checks to see if the space directly before and checks the id to see if it was a valid ptr and/or if it was freed already.