Deep Shah
Shashank Seeram
CS 214, Fall 2015
PA 3- Malloc and Free with Error Detection

The purpose of this assignment was to implement a program that stimulates a version of malloc and free with additional error checking. We used memEntry (Memory Entry) structs to keep a log of all the pointers that are malloc'd. Each Memory entry struct records if the block is free, it also checks the block size and there is a pointer that keeps track of the previous memory entry, and another pointer that keeps track of the next memory entry.

Instead of using a static char array to manage malloc and free to manage dynamic memory, we decided to use a sortedList. Since we had already created a sortedList for the last assignment, we just made some minor changes so it would work better with this assignment. We used the sortedList of memory entry structs in ascending order based on the value provided by the pointer. As we malloc'd new blocks, they kept being added to the sortedList. This made it easier to check for pointers that were never allocated memory for and for pointers that had already been freed.

The free function uses the sortedList of memory entry structs (which contains every pointer that was malloc'd) to check for already malloc'd blocks. If it doesn't exist then an error message is shown. If a block has already been freed, then an error message is shown. If the block exists, and the freeing of memory has been successful, it doesn't show anything and it tries to merge the block with the previous and next blocks if they are also free. Also, if there wasn't enough room for the new malloc'd block, we used sbrk to basically extend the memory to have enough room for the new data block and it's memory struct. The testing.c file includes the main function and that is how you would test the program.