CS 214: Systems Programming, Spring 2016
Assignment 2: A better malloc() and free()
Authors: Jordan Lyon (jjl242) and Olaolu Emmanuel (ooe4)

Implementation

- We used a linked list of “mementry” structs as descriptors for each block of memory in the heap.

Malloc: When the user asks for space, we check first for free space and sbrk if there isn't any.

Free: We look through our list of mementries for a pointer that matches the user and mark it as free.

Extra Credit Optimizations

1. **CALLOC:** We created a calloc method which calls our malloc method and then uses memset to zero-out the data.

2. **REALLOC:** We created a realloc method which calls our malloc method and then copies the users old data to the new block, using memcpy, and frees the old block.

3. **FRAGMENTATION:** To lessen fragmentation we try to lessen the amount of small blocks that are left free because they are less likely to be malloc'd for in the future. To do so, we defined a minimum size for all memory blocks. If segmenting a free block would result in making a very small block, then the entire block is allocated to the user as opposed to fragmenting it.

4. **SBRK:** As opposed to a character array, we extend the actual heap using sbrk.

5. **NO MEMORY LEAKS:** Since sbrk extends the actual heap, we created a method, cleanup(), which removes any free space from the top of the heap whenever free() is called. This way if everything is free'd, there will be no memory leaks.