Error detecting malloc() and free()

Alexander Tang, Craig Perkins

12-9-13

Implementation

For our blocks, we used two static char arrays, `big_block` and `little_block`, each of size `BLOCKSIZE` = 5000 bytes. When the user calls `malloc()`, if the data is greater than `CHUNKTHRESHOLD` = 200 bytes, it is stored in `big_block`, otherwise it is stored in `little_block`.

Case 1: free()ing pointers that were never allocated.

Check if the pointer passed to `free()`, `*p`, was within the memory addresses of our two blocks.

Case 2: free()ing pointers to dynamic memory that were not returned from malloc().

Loop through valid `MemEntry` structs in `big_block` or `little_block` and check if the pointer passed to `free()`, `*p`, matches any of the `MemEntry` structs.

Case 3: redundant free()ing of the same pointer.

Check whether or not the `MemEntry` struct corresponding to the pointer passed to `free()`, `*p`, is free. If it is, return from `free()`, display an error message, and carry on.

Case 4: Saturation.

Loop until you reach the last successor for `*p`. If `p->size < size` and `p->succ == NULL`, then you have reached the last `MemEntry` struct and there is not a chunk large enough for your data so `return NULL`, display an error message, and carry on.

Case 5: Fragmentation.

As outlined above, when the user calls `malloc()`, if the data is greater than `CHUNKTHRESHOLD` = 200 bytes, it is stored in `big_block`, otherwise it is stored in `little_block`.