Harris Seth
Von Quilon

PA6 Read Me

Welcome to the read me for the sixth assignment: Error Detecting malloc() and free(). In this assignment we apply the error checks in malloc() and free().

The program's malloc/free file has a char * array of size BLOCKSIZE, which in this case was 5000. There is a struct called mementry which is what separates out the block to give users pieces of different sizes. There were several errors that we had to account for.

Freeing pointers twice is an error . To deal with this we simply check at the beginning of the program to see if isfree() of the pointer we pass in is equal to 1 or 0. If it is equal to 1 then we have already freed this block and the user is trying to free it again. In that case we print out the error message and exit.

If the user tries to free a pointer not returned by malloc() we also print out an error message and exit. This error is detected by our mem_val array. The mem_val array is an array of the addresses of all the mementry structs in our block of memory. We add in an address to mem_val after we allocate and take out the address when we free. At the beginning of free we go through our array and check to see if any of the addresses match the address of the pointer we just passed in. If at any point the address matches we know the pointer was allocated by malloc and then we proceed to free it without any problems. If, on the other hand, we go through the array the entire array that means that we never allocated this pointer with malloc so we print the error and return.

Freeing a pointer that was never allocated is also an error. To detect this error we simply check the address of the pointer and see if it is inside of our char * block of memory. If it is not in there then we know it was never allocated, so we simply print out our error message and exit.

If all of the space been allocated, that is also an error. To detect this the program simply runs as usual, and if it makes it to the end without allocating any memory, that means that there isn't enough space to do so, and the program simply states the message and returns.

We use the __FILE__ and __LINE__ functions to tell the user in which file and line the error occurred.