DESIGN

Our approach used metadata to distinguish the blocks of allocated memory. It uses a magic number to distinguish if the metadata is valid or not, and if the metadata is valid, we are able to manipulate the memory block allocated with that metadata.

WORKLOADS

Workload 1

In this workload, we malloc one 1 byte pointer, then free it immediately. We do this 150 times.

Workload 2

In this workload, we malloc 50 1 byte pointers, then free all the pointers after. We do this process three times.

Workload 3

In this workload, we randomly choose to malloc a 1 byte pointer, or free a 1 byte pointer. Once it mallocs 50 1 byte pointers we iterate through the array of pointers freeing all of them.

Workload 4

In this workload, we randomly choose to malloc a randomly size pointer that is size between 1-64 or freeing a pointer. Tracks to make sure that the mallocs do not exceed the max size of the array. Once malloc is called 50 times it iterates through the array of pointers freeing them all.

Workload 5

In this workload, we malloc a pointer of 1000 bytes and a pointer of 5000. Should catch the the second malloc and send back an error. Then it frees both pointers. Should free the first pointer and then catch the second free and send back an error. Then it tries to free both pointers again. This time it should catch both of them and send back an error for both.

Workload 6

In this workload, we make three char pointers and one int. Mallocs the three pointers and then tries freeing address that are not the pointer and tries to free the int. Should catch these frees and send back errors. At the end it does actually free them.

FINDINGS

- We had to completely redo out methodology because we overlooked the possibility of pointers being passed that contain only some of the metadata.
- It was difficult to verify the metadata from the blocks of memory.