Systems 214
Assignment 1
Henry Chu

++Malloc

Summary:

Designing a malloc() and free() function to store memory in a simulated memory array.

Metadata:

In order to create a simulated memory allocation program, the program has to use metadata to store information about each chunk of memory - if the byte is in use and how much information each chunk stores. The program uses only 1 or 2 bytes to store metadata, with 8 bits in each byte.

The first bit stores if the chunk of memory is in use. 1 if it is, and 0 if it is not. The second bit stores how much memory the chunk stores. This is all in the first byte, with 6 bits left in the byte. However, 6 bytes can only store up to the number 63, meaning that using only one byte, the maximum size that can be stored in memory is 63 bytes. To account for this limitation, the program uses the second bit to account for numbers greater than 63. If the second bit is 1, then the program will look at the second bit for more information about how much memory is allocated for each specific chunk.

Examples:

000000 00
00 - The current chunk is not in use
000000: 0 - Amount that is allocated

000001 01
01 - The current chunk is in use and only utilizes 1 byte for metadata.
000001: 1 - Amount that is allocated

011000 01
01 - The current chunk is in use and only utilizes 1 byte for metadata.
011000: 24 - The amount that is allocated is 24 bytes.

00001111 101000 11
11 - The current chunk is in use and utilizes 2 bytes for metadata.
00001111 101000 - The amount that is allocated is 1000 bytes.

Implementation:

mymalloc:

mymalloc first starts off with the memory as one whole unused chunk. It verifies that it is empty, and allocates whatever data that is imputed in. At the same, it sets the next chunk of memory to the whole size minus the allocated size so that if more data gets allocated, it will not try to allocate in the location of memory that has already been allocated.

myfree:

With the way that the metadata is set up, if a pointer needs to be freed, there is no way to know what chunk of memory to free beforehand. myfree loops through all of the memory to determine what chunk of memory can be freed by comparing the chunk size and how much memory to free. The program then sets the matched chunk of memory to free by changing the second bit in the metadata to 0. After freeing the chunk, the program iterates through the rest of the memory and merges free blocks of memory that are next to each other.

Test Cases:

testA:

Allocates one byte and freeing it immediately 120 times
Average run time: 5 microseconds

testB:

Allocates 120 bytes individually then freeing each byte individually
Average run time: 125 microseconds

testC:

Randomly malloc or free 1 byte until 120 bytes are allocated then free all the pointers once 120 bytes are allocated
Average run time: 5300 microseconds

testD:

1. Allocate 120 bytes individually
2. Free the second fourth of allocated memory
3. Allocate 10 bytes 120 / 4 times
4. Free each pointer in the order of storage

Average run time: 180 microseconds

testE:

Allocates 120 bytes of memory then free it from the center to the end then from the start to the center

Average run time: 125 microseconds