

Similarly to the last project, some of the main challenges and obstacles in this project were the overall project design and implementation. Over the course of working on this project, the implementation progressed from a simplistic placement of each allocation to a completely working Buddy System and minorly working Slab Allocator. We started this project by implementing a linked list format. This linked list was virtually used as the actual 1MB of free memory. The nodes in this double linked list held we had structs that contained the header for the buddy system or the slab descriptor table depending on the input. In Buddy we used two of the four structs. We used node and data, node was meant to be used for the linked list and data is a part of the node struct with a struct parameter called header with the struct data type. The data struct simulated the header used in Buddy, so we knew what to free. In Slab we used the remaining structs s\_node and slab. The struct slab is our implementation of the Slab Description Table which was a linked list of slabs pointing to one another. The s\_nodes structs have the slab struct's parameters and it was a linked list of the objects that point to one another. We attempted to make our my\_malloc and my\_free work in a similar manner to our buddy version, but we ran into problems. Similar in the sense of our methods of keep track of block, slab, and object sizes and how we kept track of the starting and ending addresses of each mentioned respectively. These objects acted as the physical allocation spots for each object passed into my\_malloc.

Using a double linked list implementation to set up both the Buddy System and the Slab Allocator allowed for us to manipulate and move portions of the allocated memory without having to actually manipulate the physical memory whatsoever. Our Buddy System implementation works with power of 2 math to allocate blocks based on the Buddy System allocation algorithm. Our Slab Allocator implementation uses the Buddy System to implement the slabs and then places each object of that type one after another in the slab until it is either full or no other objects remain to be placed inside that slab.

For this project we changed up how we did the breakdown and overall method of work for each of us. In the last project we attempted to work together as much as possible and did not do as much independent work. For this project we broke down what needed to be done at the very beginning and split the work of setting up the base implementation. Yonathan initially worked on getting the linked list implementation to work correctly, and Gabien initially setup the overall structure for using the Buddy System and freed the memory with the Buddy System implementation. Splitting the work in the beginning proved very beneficial as we easily merged our work on the double linked list and the Buddy System. Once the code was merged and organized, we split off to independently debug and attempt to get the Buddy System working correctly. The rest of the work done on the project was a majority of each student committing their code after any sort of breakthrough in functionality. By doing this we were able to consistently build off each other's commits and continue to make progress or improvements.

Some additional information that we wanted to include is that when running the Slab Allocator on the Ubuntu terminal we were able to get correct outputs for various of the inputs, but when we ran the Slab Allocator on the w204 machines, we encountered various errors in the output that was not evident on the Ubuntu vm. A picture of the output for the Slab Allocator ran on the

Ubuntu terminal can be found at the end of this document. For example in the picture provided, Ubuntu allows for the allocation of object A, but on the w204 machines we consistently got an error where object A was not allocated. This seemed to be the major error between the two terminals. On an Ubuntu 64-bit virtual machine, we very rarely experienced errors in the actual allocation of memory. On the w204 machines, we frequently received errors in allocation for various objects.

```
gablenbryan@ubuntu:~/Desktop/p2-2020-memory-allocation-yonathan-and-gablen$ ./out 1 TestInputs/input_1
Start of first Chunk Z is: 8
Start of first Chunk Z is: 1246
Start of first Chunk Z is: 2484
Start of first Chunk Z is: 3722
Start of first Chunk Z is: 4960
Start of Chunk A is: 524296
Start of Chunk C is: 131080
Start of Chunk C is: 131103
Start of Chunk C is: 131126
Start of Chunk C is: 131149
Start of Chunk C is: 131172
Start of Chunk C is: 131195
Start of Chunk C is: 131218
Start of Chunk C is: 131241
Start of Chunk C is: 131264
Start of Chunk C is: 131287
Start of Chunk C is: 131310
Start of Chunk C is: 131333
Start of Chunk C is: 131356
Start of Chunk C is: 131379
Start of Chunk C is: 131402
Start of Chunk C is: 131425
Start of Chunk C is: 131448
Start of Chunk C is: 131471
Start of Chunk C is: 131494
Start of Chunk C is: 131517
freed object Z at 8
freed object Z at 1246
freed object Z at 2484
freed object Z at 3722
freed object Z at 4960
Start of Chunk U is: 262152
Start of Chunk U is: 263386
Start of Chunk U is: 264620
Start of Chunk U is: 265854
Start of Chunk U is: 267088
Start of Chunk V is: 139272
Start of Chunk V is: 139399
Start of Chunk V is: 139526
Start of Chunk V is: 139653
Start of Chunk V is: 139780
Start of Chunk V is: 139907
gablenbryan@ubuntu:~/Desktop/p2-2020-memory-allocation-yonathan-and-gablen$
```

```
Start of first Chunk A is: 498216
Start of first Chunk A is: 498520
Start of first Chunk A is: 498824
Start of first Chunk A is: 499128
Start of first Chunk A is: 499432
Start of first Chunk A is: 499736
Start of first Chunk A is: 500040
Start of first Chunk A is: 500344
Start of first Chunk A is: 500648
Start of first Chunk A is: 500952
Start of first Chunk A is: 501256
Start of first Chunk A is: 501560
Start of first Chunk A is: 501864
Start of first Chunk A is: 502168
Start of first Chunk A is: 502472
Start of first Chunk A is: 502776
Start of first Chunk A is: 503080
Start of first Chunk A is: 503384
Start of first Chunk A is: 503688
Start of first Chunk A is: 503992
Start of first Chunk A is: 504296
Start of first Chunk A is: 504600
Start of first Chunk A is: 504904
Start of first Chunk A is: 505208
Start of first Chunk A is: 505512
Start of first Chunk A is: 505816
Start of first Chunk A is: 506120
Start of first Chunk A is: 506424
Start of first Chunk A is: 506728
Start of first Chunk A is: 507032
Start of first Chunk A is: 507336
Start of first Chunk A is: 507640
Start of first Chunk A is: 507944
Start of first Chunk A is: 508248
Start of first Chunk A is: 508552
Start of first Chunk A is: 508856
Start of first Chunk A is: 509160
Start of first Chunk A is: 509464
Start of first Chunk A is: 509768
Start of first Chunk A is: 510072
Start of first Chunk A is: 510376
Start of first Chunk A is: 510680
freed object A at 8
Start of Chunk B is: 524296
Start of Chunk C is: 655368
gabienbryan@ubuntu:~/Desktop/p2-2020-memory-allocation-yonathan-and-gabien$
```

We did get the Buddy Allocation System to work for all the test inputs given to us, but the Slab Allocation System was not correct. We do get large portions of correct output, but it is not one hundred percent correct.