In this project, Dylan and I worked to implement and compare memory allocation/deallocation schemes, specifically First Fit, Best Fit, Worst Fit and Buddy system. One of the jobs of the memory management subsystem is to service memory requests by matching the size of the request with a large enough hole, from which to satisfy the request. In a file called my\_memory.c , we implemented the following functions that perform allocation/deallocation:

**void setup(int malloc\_type, int mem\_size, void\* start\_of\_memory);**

The first argument specifies the type of memory allocation (0 for First-Fit, 1 for Best-Fit, 2 for Worst-Fit and 3 for Buddy System). The second argument specifies how much total memory space is available in the system (1 MB). The third argument specifies the pointer to the start of this memory space, from which all allocations using my\_malloc() should be made. The purpose of setup () is to perform any initialization of variables that you may need, specify and give you the pointer to the total amount of memory at your disposal, and also specify the type of memory allocator.

**void \*my\_malloc(int size);**This allocates memory segments from start\_of\_memory using the specified allocation algorithm.

**void my\_free(void \*ptr);**That deallocates the memory segment being passed by the pointer. When free-ing, the resulting free segment will be merged with relevant neighboring holes, depending on the scheme declared, whenever possible to create holes of larger size.

**Design Choices**

To keep track of memory we used a doubly linked list of memory nodes. We used a doubly linked list to keep inserts, and deletes simple and fast. Each memory node in the list contains 2 pointers, start and size. Size pointer will always point to the start pointer and will contain the integer size of the memory node. All start pointers are relative to the start\_of\_memory address. Each memory node also contains a used flag to determine whether or not that node is allocated or free. When returning from my\_malloc we return the free memory node’s start pointer + 4 and when freeing, we go back 4 bytes (ptr-4) to get the size of the memory node we want to free.

Contributions

Dylan Nguyen: First Fit and Buddy System schemes and my\_free

Jacob Klapper: Best Fit and Worst fit schemes, additional test cases