Asst 1: A Better Malloc

Colin Ackerley & Warren Ho

**Design:**

Our design has two blocks of metadata. The metadata is contained within two bytes. The most significant bit is used to store whether or not a given block is allocated. 0 if it is available, 1 if it is allocated. The remaining 15 bits are used to store the size of the following block. Since we are using all unsigned values, we will have 2^15 as our maximum size, which is more than enough for our purposes.

The 0th block is initially set to a size of 5000, and then it is shrunk as the user begins requesting space. This approach works very well for what we need it to do. We eliminated the need for an extra structure to store information. It does lower the amount of user accessible memory, but two bytes of metadata per block the user creates is reasonable. It is also very fast in terms of efficiency, as all of the reading is done via bitshifts. Using a separate data structure would have been detrimental in terms of performance.

In simple terms, our structure is blocks of data that are either allocated or not allocated. The blocks will merge together if they are freed and next to each other, and they will shrink as they are allocated.

MyMalloc first checks some basic error conditions. It then looks for a free block. If it is able to find one, then it will mark the block as allocated. If there is space left in the block, it will adjust the size of the non-allocated block accordingly.

MyFree first does some basic error checking. It will then locate the address the user is trying to free. If it has already been freed (allocation bit is 0), it will give the user an error. Otherwise, it will mark the bit as not allocated. It will then perform checks to see if adjacent blocks are free, and if they are, it will merge them together.

**Test Case Results & Observations:**

Case A: 0.006525

Case B: 0.006311

Case C: 0.005546

Case D: 0.005888

Case E: 0.005661

Case F: 0.031771

Our code is mostly working. Our malloc works down to the byte. Our free logic is also working. If our two methods for set and get size worked, then all of our code would work perfectly. I believe that we are losing the allocation bit. The allocation bit is the most significant bit, and the remaining 15 bits are our size.