David Wang, Mark Vaysiberg

Brief Description

Our metadata consists of a short for the size of the block (blockSize) and a character to indicate whether it's in use or not (inUse). We chose to use a short so it would save more space compared to an int. The max size of our memory is 4096 and this size falls within a short. We also decided to use a character because it only takes up 1 byte which is the least amount of space compared to other data types. In order to access the next block's metadata, we increment the pointer by the size of the metadata and block with our nextMeta function. In order to access the pointer to the data block, we increment the pointer to inUse by one.

Error Handling

- myMalloc: If there is not enough space to malloc the user's requested size or if the user requests a size of 0, an error message detailing the error with the file and line number where it occurred is printed.
- myFree: If the pointer provided by the user does not point to the beginning of a data block, an error message with the file and line number where it occurred is printed.

Efficiency

- To insert data, we traversed the memory until an unused block which has space >= size was found or the end of memory was reached. This has the efficiency of O(n). Creating the metadata with the data block was done with our insertData function which calls insertMetadata.
- To free data, we traversed the memory to keep track of the previous metadata until the pointer was found or the end of memory was reached. This also has the efficiency of O(n). The previous pointer checks whether it is unused so it can be merged with the freed block.
- Space is O(1) because our memory size is constant.

Great Features

- If the size of the block where the new memory will be located does not have enough space for a new metadata block and free space, we give the entire block to the user as we don't want to leave space that doesn't have metadata.
- If there are unused blocks adjacent to the freed block, they are merged together.