1. Metadata
    a. Metadata stores information in 3 bytes describing the space that mymalloc has allocated for the user. The first byte stores a character value that represents whether the space is free or not. The next two bytes store a short integer value that represents the number of bytes that the space takes up.
2. Allocated space
    a. Allocated space belongs immediately after the metadata. If a mymalloc call is successful, the program will return a pointer to the first byte of some space that the user may use.
3. Array
    a. The array stores no information until the program calls mymalloc for the first time. After the program calls mymalloc for the first time, the array will have the following characteristics:
        i. Each element of the array belongs to either metadata or allocated space
        ii. The first 3 bytes of the array must consist of metadata
        iii. The array may have more than one metadata
        iv. At least one byte of allocated space must follow each metadata
4. Mymalloc Overview
    a. If a mymalloc call is successful, it will return a pointer at allocated space to the user, and update the metadata located immediately before the space to store the following information:
        i. That the space is not free
        ii. The size of the space
    b. The mymalloc call will create metadata at the end of the space depending on whether there is space of another metadata and at least one byte of additional space after that.
5. Possible Errors that Mymalloc May Catch
    a. Mymalloc will print information about the following errors and return null if it was not successful:
        i. The input size is not a positive number
        ii. The array did not have enough space to return the space that the user requested and a metadata to describe the space
6. Myfree Overview
    a. If a myfree call is successful, it will free a space that had been allocated to the user. If the space after it was also free, the myfree call would merge the two blocks so that they could become one large free block. If the space before it was also free, the free call would merge the two blocks as well.
7. Possible Errors that Myfree May Catch
    a. Myfree will print information about the following errors if it was not successful:
        i. The parameter was null
        ii. The parameter was not a pointer that was returned by the mymalloc function
        iii. The parameter was a pointer to space that was already free
8. Mymalloc Algorithm Description
    a. The mymalloc method takes one parameter: the number of bytes of space to allocate. A NULL pointer and an error message will be returned if the size

requested is beyond 1-4096. When mymalloc is called for the first time (identified by the value casted as an integer in the second byte of the static array being 0), it initializes the array with the first metadata by setting the first byte as “free” and setting the next two bytes as 4093 to represent the initial data block size. A NULL pointer will be returned if the size requested is more than 4093.After the first initialization, mymalloc does one of three things. First, if the current memory block is not free or does not have the available space for the requested number of bytes, then mymalloc moves onto the next block. Second, if it does have the space for the requested number of bytes plus 0-3 additional bytes, it will return a pointer to that space in the current memory block without creating a new metadata. Third, if the number of bytes in the memory block is greater than the number of bytes requested by 4 or more (enough for one metadata and at least one byte for data), then a new metadata is created for a new free block after the current block and a pointer to the current block right after the current metadata will be returned.

9. myfree Algorithm Description
    a. The myfree method will check to see if its parameter is a pointer in the range of the character array. If it the pointer is not, the myfree method will output the appropriate error message. Next, the myfree method will traverse the character array to check all of the possible spots that a pointer may be at, which are the spots that any metadata ends. If the pointer is not at those spots, then the method will output an error message stating that the parameter was not a pointer that was allocated by mymalloc. If the pointer is at those spots, the method will check if the space that the pointer is pointing to is labelled as free or not. If the pointer is pointing at space that is free, then the method will output an error message that states that the pointer location has already been freed. If the pointer is pointing at a space that is not free, then the method will label the space as free. The method will check if the spaces after or before this space are free. If so, the method will merge the two or three free blocks to form one block.
10. memgrind Description
    a. From running memgrind and finding the mean run times, we can see how fast each of the workloads is.
    mean run time for A (in 100 runs) : 2 (microseconds)
    mean run time for B (in 100 runs) : 22 (microseconds)
    mean run time for C (in 100 runs) : 4 (microseconds)
    mean run time for D (in 100 runs) : 5 (microseconds)
    mean run time for E (in 100 runs) : 34 (microseconds)
    mean run time for F (in 100 runs) : 39 (microseconds)
    A, C and D take up a nearly insignificant amount of time. The mean time of workload A is usually 1-10 microseconds. The mean times of workloads C and D are usually around 3-5 microseconds. This is probably because A, C, D, regardless of the length of code, run loops that are much shorter and less frequent than the other workloads. For example, A only has one while loop, while B has two while loops. Thus, B has longer mean times, which are around 20 microseconds. Workload E has a mean time of around 33 microseconds, when it has 3 while loops and a for loop. Finally, workload F tends to take the longest amount of time, near 50 microseconds. This makes sense with the previous

observations, because each of the four loops in workload F has to run through around 100 or 50 elements, meaning it would take longer than the other workloads.