Christie Chang (cc1173)
Samuel Yang (sy369)

Assignment 1

Design:

mymalloc()
This function has a char array myblock[5000] that holds values to indicate whether the space is taken ('1') or free ('0').
mymalloc() will first determine whether there is enough space for the given size to be inputted into the array.
If the inputted size is avaialble, then mymalloc() will allot the space for the given size, update myblock to '1', and return the first memory address of the space.
If the inputted size is not available, then mymalloc() will return 0.

myfree()
This function checks to see if there is stored memory at the given address.
If the address is not in the set of addresses for the array, then an error is returned.
If the address does not have a pointer stored (for example, a pointer already freed), then an error is returned.
If the address does have a pointer stored, then the pointer will be freed and the value in myblock is changed to '0'.

checkspace()
This function checks myblock to see if a given size is available in the array.
True is returned if the space is available and False is returned if the space is not available.

memcheck()
This function was used in the debugging process to

determine whether the array was storing information properly by outputting the 0's and 1's in myblock.

Workload:
mean time for execution of test A: 0.019847 seconds

mean time for execution of test B: 0.000082 seconds

mean time for execution of test C: 0.006812 seconds

mean time for execution of test D: 0.026914 seconds

mean time for execution of test E: 0.010571 seconds

mean time for execution of test F: 0.000133 seconds

Findings:
We found that our malloc and free functions increase runtime when ran separately (test A). This is due to our check function, which runs in O(n) time. Since the array gets filled to 3000, the runtime for malloc-ing 3000 times would be 0(n^2). As you can see in test B, when only the first block in the array is used, the run time significantly faster since it is running in O(n) time.
We also found that our program is very inefficient because our memory doesn't defragment, thus it spends more time looking for free space at the end of the myblock memory array.