**Assignment 1**

Kapil Jayaraman- kj184

Hiren Patel- hkp35

**Usage:**

Just run the make and execute ‘memgrind’ in order to test the requirements A through F.

**Description:**

This is our implementation of the malloc() and free() library calls. The benefit to our implementation is that it allows us to detect common dynamic memory errors. We use a static char array of size 5000 to manage the dynamic memory so that we don't have to worry about where our memory actually comes from. All cells in the array are initialized to 0 (NULL). The purpose for this is to keep track of legitimately malloced memory space and is used in detecting a lot of the errors. When you malloc(), a block of space is given based on the requested size. Right before the block of space is the node struct header lets us handle how much space is used and/or available. When you free() it starts from the head and checks each beginning pointer to node \* -1 the block is freed and merged with nearby free blocks. If the pointer is invalid, an error is returned.

**Errors Caught:**

* Freeing NULL pointers
* Freeing pointers that were never allocated
* Freeing pointers redundantly (freeing when already freed.)
* Allocating for a block the size of 0 or less bytes
* Saturation - Allocating for a block when not enough memory space available
* Fragmentation – The fix is to increase maximum amount of dynamic memory available

We found that: for example there is a block of 1500 and we malloc 1000 then free the first 500 and try to malloc 700 our implementation of malloc will state insufficient memory.

**Timings:**

The timing averages we saw was around what we were expecting the values to be around when we averaged the 100 runs.