Asst1 - ++Malloc

Karun Kanda (kk951), Junxian Cai (jc2411)

October 25, 2020

1 Name of Programs

mymalloc.c - implements the mymalloc function and myfree function.
mymalloc.h - contains the metadata struct, utilized macros and function prototypes.
memgrind.c - runs the workloads to test mymalloc and myfree.

2 Usage

```
./memgrind
```

3 Design Features of mymalloc and myfree

When implementing mymalloc and myfree the main purpose of these two functions are to error check what malloc and free wouldn't check so keeping in that mind.
The mymalloc function takes the requested byte size, the file that the user is allocating data in and the line number of where the request took place and returns a void pointer based on the requested byte size:

```
void *mymalloc(size_t size, const char *file, int line);
```

The main differences between mymalloc and malloc are:

1. if the size is larger than the total bytes in the MemoryBlock - the block size of the metadata
2. Checks if the block that is being allocated is within the MemoryBlock
3. The block that is being allocated is less than the requested size of allocation

For the myfree function is takes a void pointer, file name and the line number and frees the pointer from memory which in the case of the assignment is in

MemoryBlock.

```
void myfree(void *p, const char *file, int line);
```

The one main difference between myfree and free is:

1. After myfree frees the memory it will coalesce adjacent free blocks and find the correct size for the free block.

4 Metadata Design

For our metadata design we decided to make a struct that holds the size that was requested by the user, a free variable that indicates whatever the block is free or not and a pointer to the next point in memory.

```
typedef struct MetaData
{
    size_t size;
    unsigned short free;
    struct MetaData * next;
} MyBlock;
```

The design was envisioned to hold this metadata and at the end the amount the user wants to store in memory. The total size of the metadata is 8 (size_t size) + 2 (unsigned short free) + 8 (struct MetaData * next) + 6 (for the additional padding) making the total 24 bytes.
We designed the metadata to go in front of allocated bytes the user requested and chained that in a linked list type implementation where the next node would be another memory block.