Microcontrollers and Microprocessors

Assignment-1

**Please Note:**

I have used the logic/functions present in some of these websites for guidance as I could not get my own functions to work before the deadline.

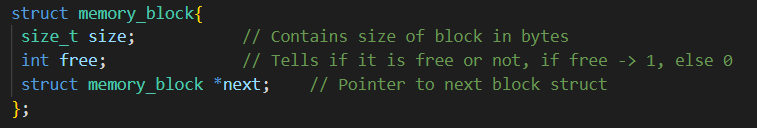
Website Links:

1. <https://stackoverflow.com/questions/13764711/making-your-own-malloc-function-in-c/63745996#63745996>
2. <https://github.com/sonugiri1043/Malloc-Free/blob/master/myMalloc.c>
3. <https://danluu.com/malloc-tutorial/>

**Data structure used:**

I have used a linked list kind of data structure to keep track of the metadata of each memory\_block struct and the space in memory. Every chunk of memory block has metadata attached to it… through this structure, we can find out the next unused block of free space and also find out how much free space is present in that memory\_block.

The struct contains three values:



Size, free (tells if memory\_block is free or not) and next (pointer to next struct value)

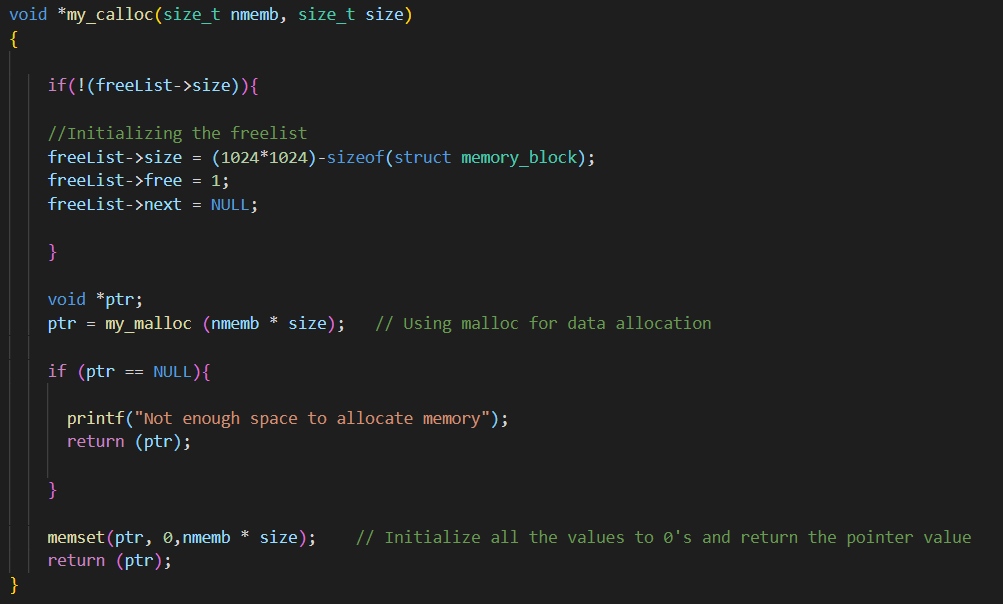
**Full explanation of algorithm used:**

**My\_malloc() Implementation:**

When we call my\_malloc() function with its specified parameters, first we take the memory and make a freeList. Suppose we want a space of 500 bytes, we go through this list of memory\_block structs,

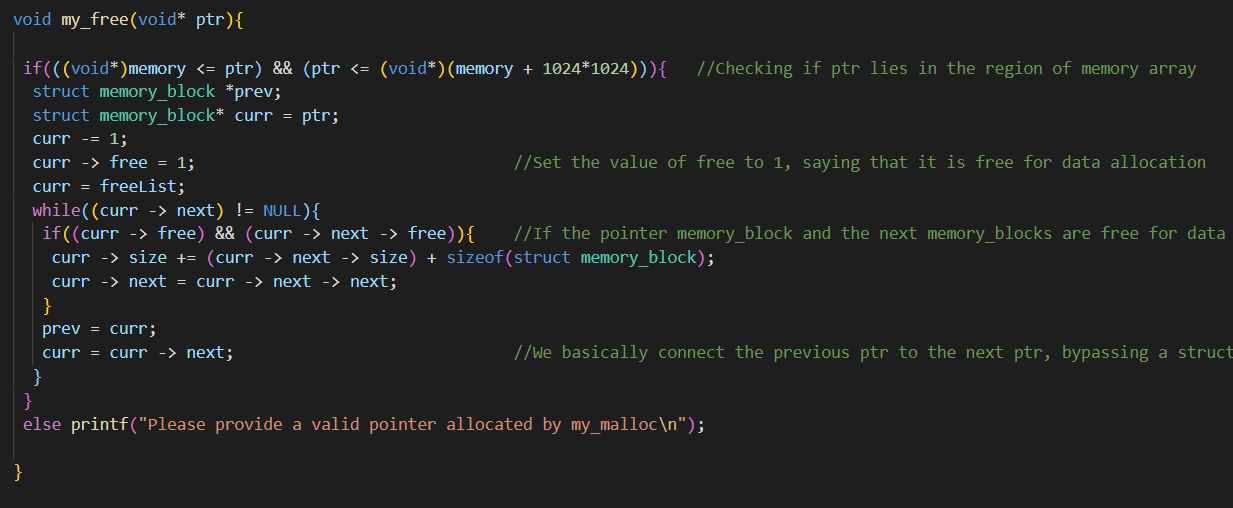
1. If we find memory\_blocks which are used (or) have lesser space than what is asked (500 bytes in our case), then we ignore the block and move on to the next memory\_block in the list.
2. If we find a memory\_block whose size is exactly what we need, we will allocate our data there and return the pointer to that position as result
3. If we find a memory\_block which has more size than what we need, we will split the memory\_block into two smaller memory\_blocks such that one block has the size we exactly need and the other block has the remaining bytes, (ie, size of memory\_block – required size)

**My\_calloc() Implementation:**



As we have already implemented malloc(), we can take advantage of it and use malloc() function along with bzeros() or memset() function (To set the uninitialized values to zeros) to get the required result of calloc() function.

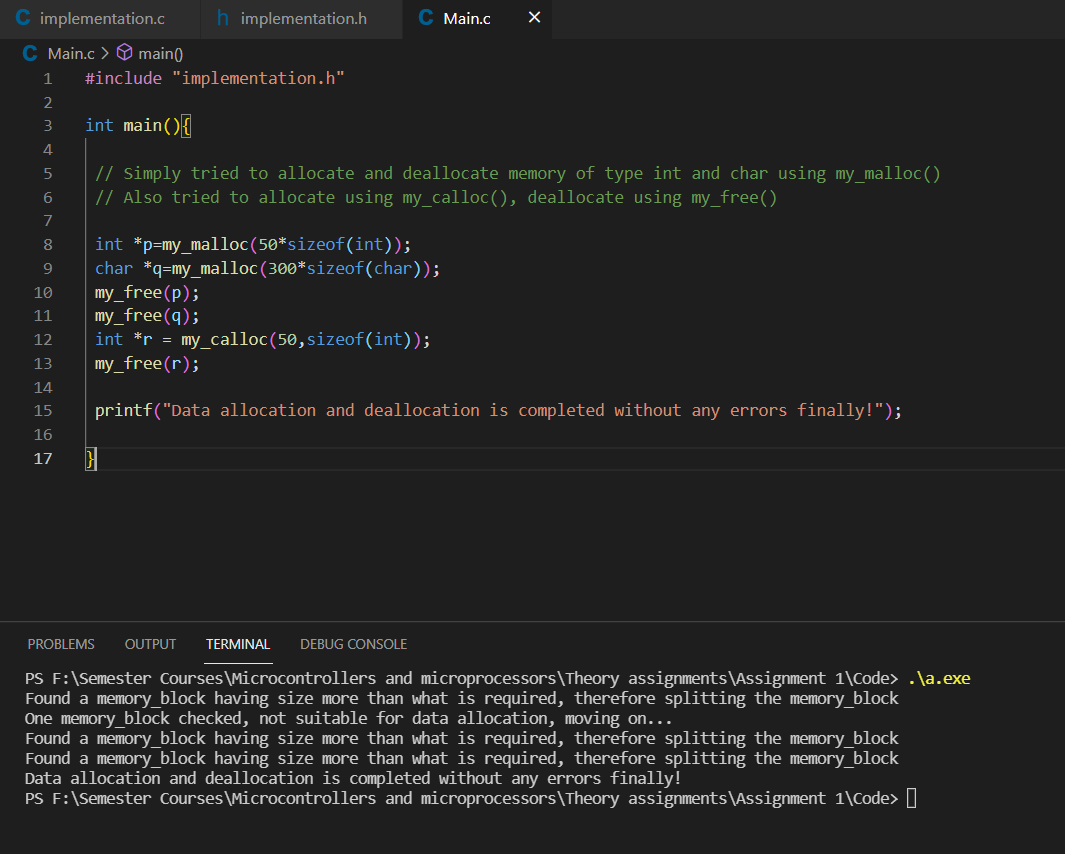
**My\_free() Implementation:**

****

First we check if the pointer passed to my\_free() function as parameter lies in our memory array or not. Value of pointer should be between memory pointer and (memory + memory\_size) pointer. In our case, we have taken memory\_size to be 1024\*1024 bytes. After checking we go to the previous memory\_block of the provided pointer, set the block’s free value to 1 stating that the memory\_block it is pointing to is free now. We also set its next value to the next next block value of the provided pointer.

Please refer to the code file, I have made sure to comment it thoroughly as far as to what I could understand.

**Output Results (Screenshot):**



The Main.c, implementation.c and implementation.h files are present in the code folder which is submitted along with this design document.