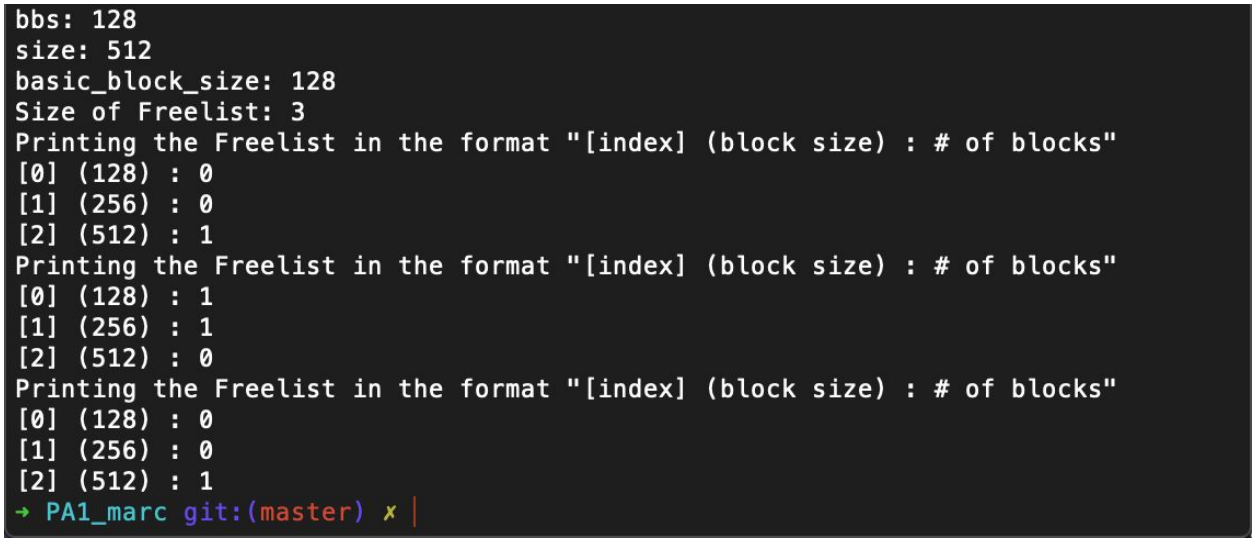
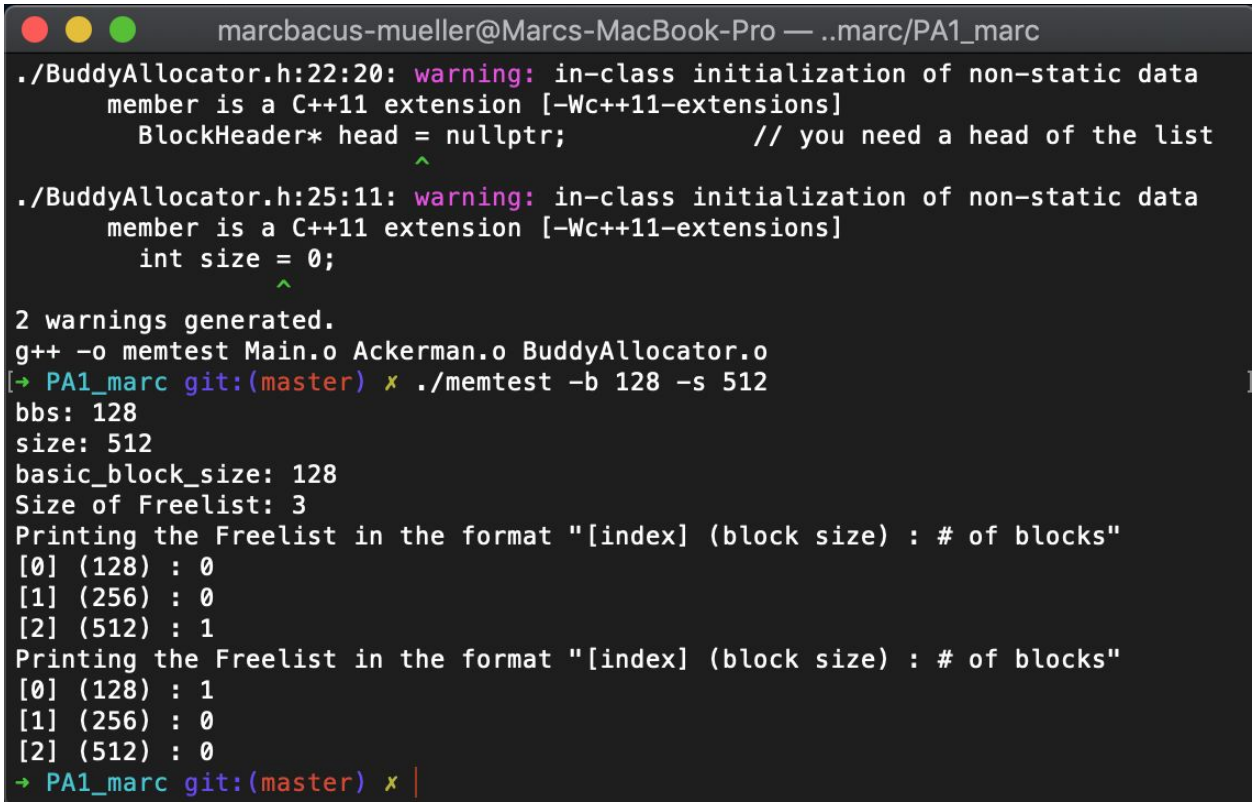
Programming Assignment 1

Marc Bacus-Mueller
Friday September 20 , 2019
CSCE 313
Dr.Tanzir Ahmed

In this report I will be proving that my individual functions worked and my ackermann function graphs. I will also be discussing bottlenecks and how this implementation could be improved. As far as the easy test, my program was able to handle allocating memory, splitting up memory, freeing it, and finally merging it. These are a couple screenshots of the code working in my terminal.



After the mem was freed, it was then merged back into a big 512 block.
To prove that my split function was working, here is what happens when I want to allocate 3 pieces of 1 byte memory,

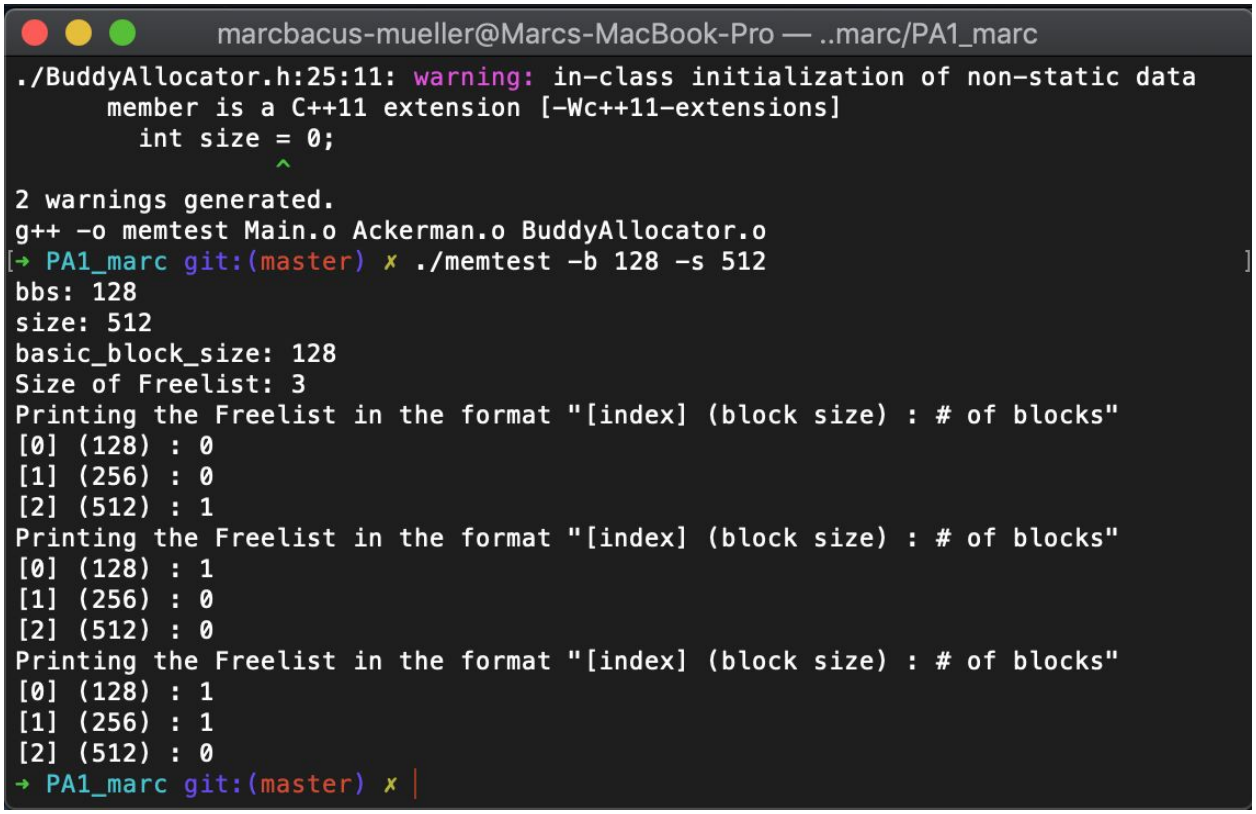


As you can see, the 256 block was forced to split up resulting in the last remaining 128 block.

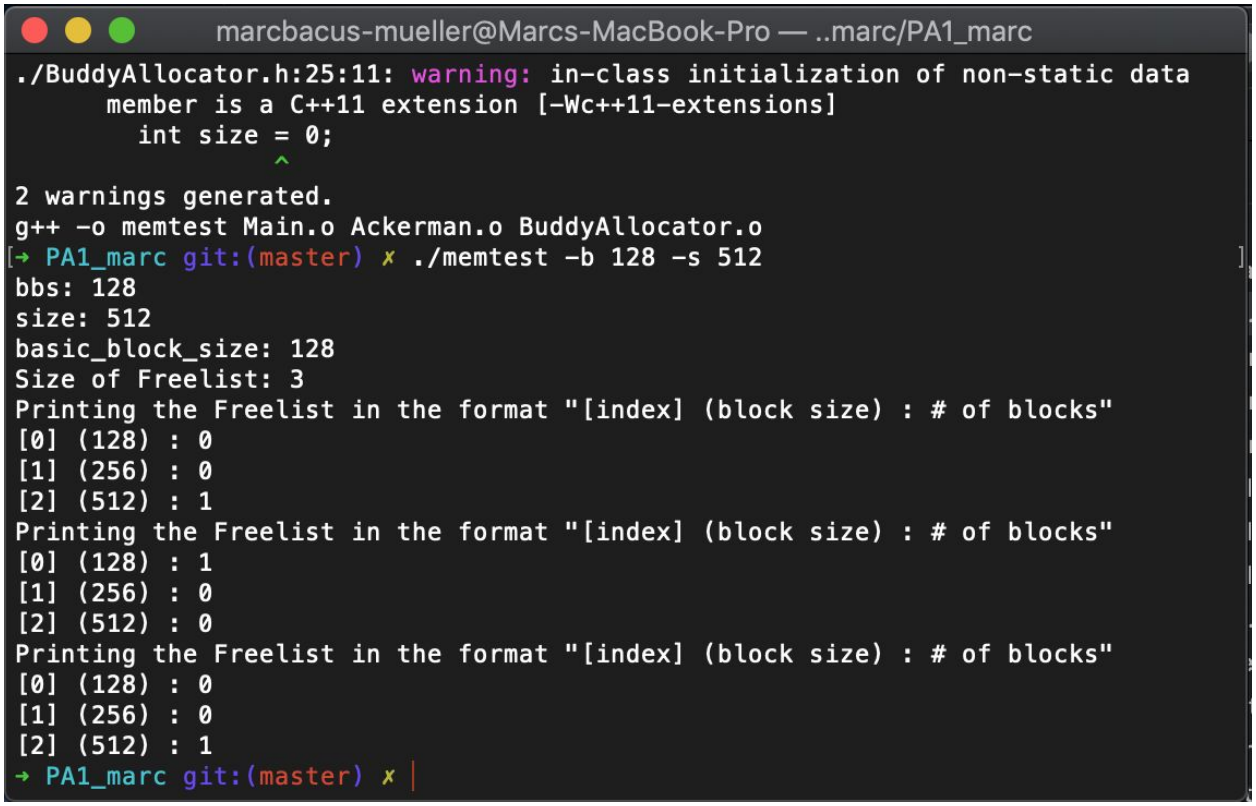
My getbuddy and free function also worked as well. Here is a screenshot of me allocating 3 pieces of memory, and only freeing 2 pieces of them. Here is the code that I will be sampling.

```
ba->printlist();
// allocating a byte
char * mem = ba->alloc (1);
  char * mem2 = ba->alloc (1);
  char * mem3 = ba->alloc (1);

// now print again, how should the list look now
ba->printlist();

ba->free (mem2);// give back the memory you just allocated
  ba->free (mem);

  //ba->free(mem9);
ba->printlist(); // shouldn't the list now look like as in the beginning
```

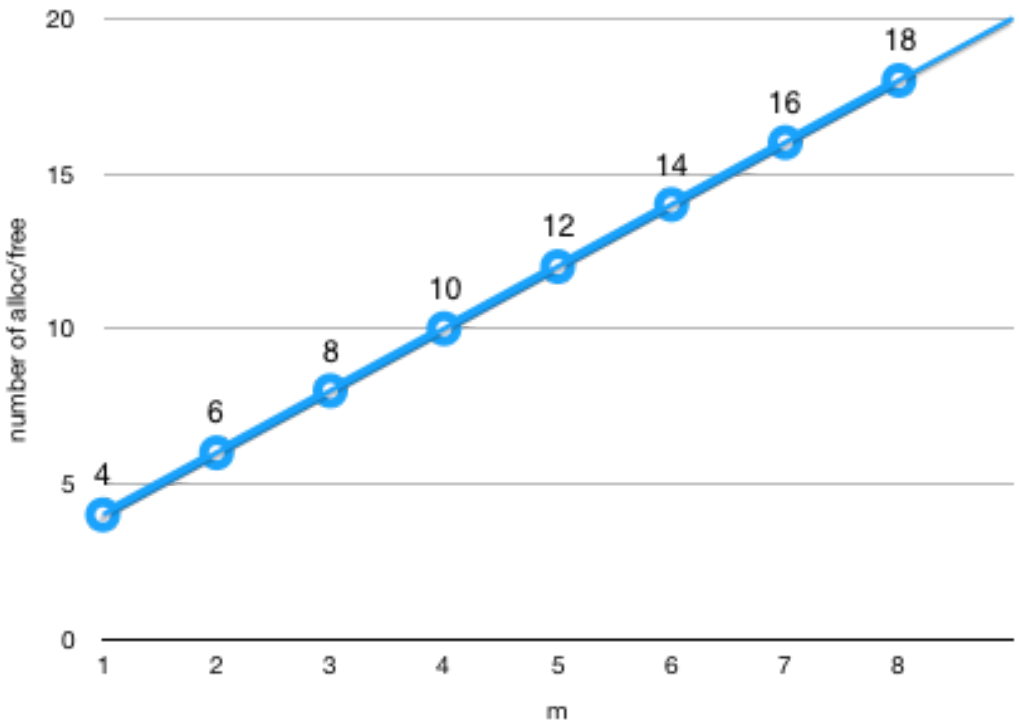
And here are the results



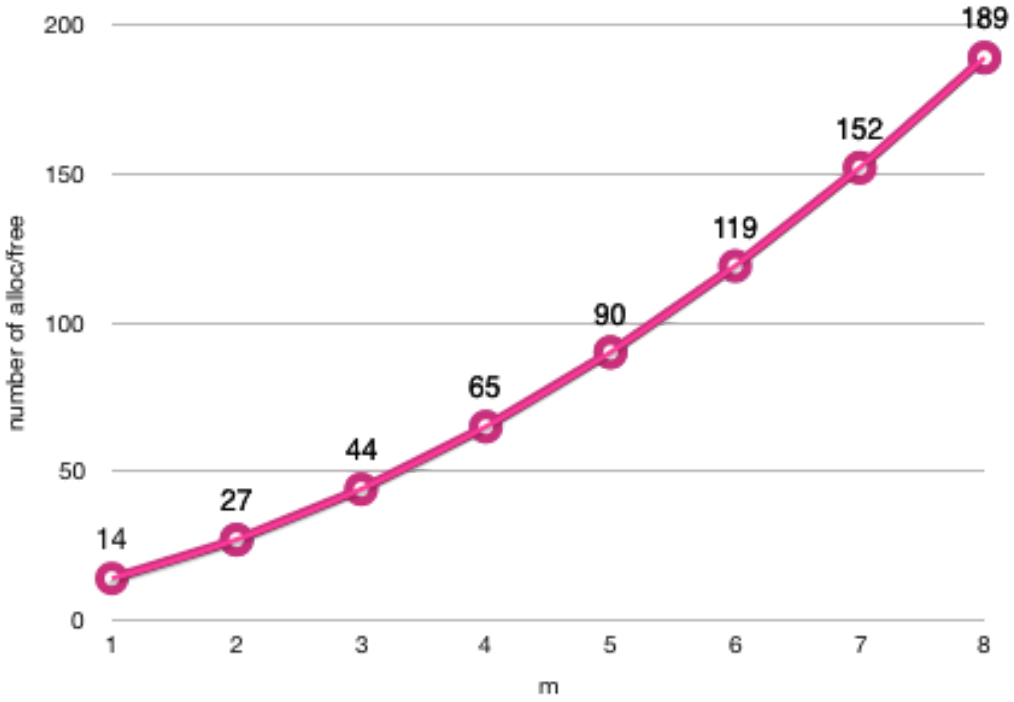
As you can see, a mem and mem 2 were buddies that merged together to form the 256 block. This means that if I were to free mem3, everything would merge together with their buddies and create a 512 block. Here is that test.



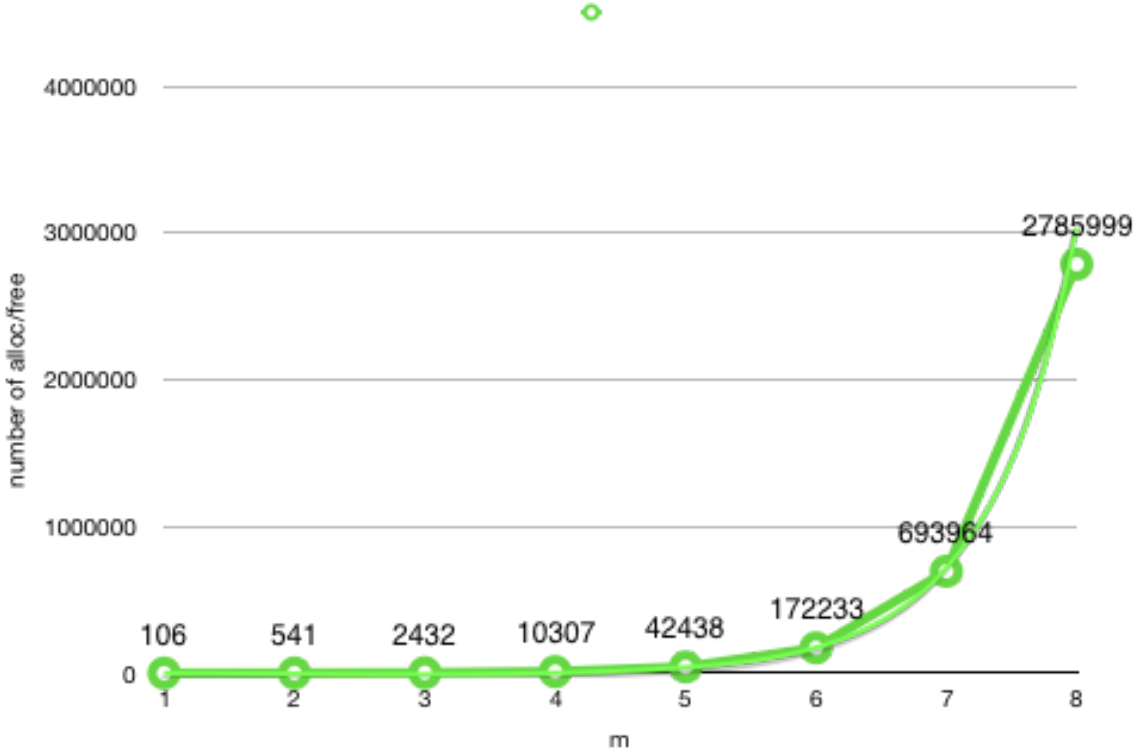
Now onto Ackermann, the following pictures are my graphs for each n = (1,2,3) and the x axis stands for the m = (1,2,3,4,5,6,7,8). I will describe my observation under each photo. All these tests are on the basis that the bbs is 128 b and the total memory is 134217728b.



As you can see, for when n is 1, the line of best fit represents a linear line when it comes to the number of allocations and frees.



As you can, for when n is 2, the line of best fit represents a polynomial equation when it comes to the number of allocations and frees.



And last, for when n is 3, the line of best fit represents most similarly to an exponential graph when it comes to the number of allocations and frees.

One bottleneck that I suspect would be the merge has to go through each level in the program which could take a long time if we had more memory and a small basic block size. This is the same case for alloc where if the first index does not work, my program has to go through the entire index of the freelist until it finds one that is free. This is definitely a bottleneck that affects the time of the program itself. A solution could be some sort of secondary vector named avaiable_blocks that knows all the possible blocks that the program can automatically jump to without having to check each index.