PA1 Report by Siyuan Yang

1) Test different ack (m, n) combinations.

```
==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 1
Ackerman(1, 1): 3
Time taken:  [sec = 0, musec = 978]
Number of allocate/free cycles: 4

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 2
Ackerman(1, 2): 4
Time taken:  [sec = 0, musec = 998]
Number of allocate/free cycles: 6

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 3
Ackerman(1, 3): 5
Time taken:  [sec = 0, musec = 999]
Number of allocate/free cycles: 8


==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 4
Ackerman(1, 4): 6
Time taken:  [sec = 0, musec = 1000]
Number of allocate/free cycles: 10
```

```
==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 5
Ackerman(1, 5): 7
Time taken:  [sec = 0, musec = 1025]
Number of allocate/free cycles: 12

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 6
Ackerman(1, 6): 8
Time taken:  [sec = 0, musec = 1027]
Number of allocate/free cycles: 14

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 7
Ackerman(1, 7): 9
Time taken:  [sec = 0, musec = 2017]
Number of allocate/free cycles: 16

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 1
  m = 8
Ackerman(1, 8): 10
Time taken:  [sec = 0, musec = 2992]
Number of allocate/free cycles: 18
```

```
==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 1
Ackerman(2, 1): 5
Time taken:  [sec = 0, musec = 1966]
Number of allocate/free cycles: 14

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 2
Ackerman(2, 2): 7
Time taken:  [sec = 0, musec = 3978]
Number of allocate/free cycles: 27

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 3
Ackerman(2, 3): 9
Time taken:  [sec = 0, musec = 4016]
Number of allocate/free cycles: 44

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 4
Ackerman(2, 4): 11
Time taken:  [sec = 0, musec = 4949]
Number of allocate/free cycles: 65
```

```
================================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 5
Ackerman(2, 5): 13
Time taken:  [sec = 0, musec = 7973]
Number of allocate/free cycles: 90

================================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 6
Ackerman(2, 6): 15
Time taken:  [sec = 0, musec = 8976]
Number of allocate/free cycles: 119


================================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 7
Ackerman(2, 7): 17
Time taken:  [sec = 0, musec = 12963]
Number of allocate/free cycles: 152

================================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 2
  m = 8
Ackerman(2, 8): 19
Time taken:  [sec = 0, musec = 16978]
Number of allocate/free cycles: 189
```

```
==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 1
Ackerman(3, 1): 13
Time taken:  [sec = 0, musec = 5974]
Number of allocate/free cycles: 106

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 2
Ackerman(3, 2): 29
Time taken:  [sec = 0, musec = 37888]
Number of allocate/free cycles: 541

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 3
Ackerman(3, 3): 61
Time taken:  [sec = 0, musec = 160571]
Number of allocate/free cycles: 2432

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 4
Ackerman(3, 4): 125
Time taken:  [sec = 0, musec = 669195]
Number of allocate/free cycles: 10307

==========================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 5
Ackerman(3, 5): 253
Time taken:  [sec = 2, musec = 948127]
Number of allocate/free cycles: 42438
```

2) Test ack (3,6) 3 consecutive times with bs=128 and m=64MB

```
PS C:\Users\SuperSteven\Desktop\CSCE 313\Lecture\PA\PA1\Code> ./memtest -b 128 -s 67108864
==================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 6
Ackerman(3, 6): 509
Time taken:  [sec = 12, musec = 329073]
Number of allocate/free cycles: 172233

==================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 6
Ackerman(3, 6): 509
Time taken:  [sec = 11, musec = 711713]
Number of allocate/free cycles: 172233

==================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 6
Ackerman(3, 6): 509
Time taken:  [sec = 11, musec = 438447]
Number of allocate/free cycles: 172233
```

3) Test ack (3,8)

```
PS C:\Users\SuperSteven\Desktop\CSCE 313\Lecture\PA\PA1\Code> ./memtest -b 128 -s 67108864
==================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 8
Ackerman(3, 8): 1582
Time taken:  [sec = 173, musec = 260758]
Number of allocate/free cycles: 2570620
```

4) Run ack (3,6) with total memory = 128KB

```
PS C:\Users\SuperSteven\Desktop\CSCE 313\Lecture\PA\PA1\Code> ./memtest -b 128 -s 131072
==================================================================
Please enter parameters n (<=3) and m (<=8) to ackerman function
Enter 0 for either n or m in order to exit.

  n = 3
  m = 6
PS C:\Users\SuperSteven\Desktop\CSCE 313\Lecture\PA\PA1\Code>
```

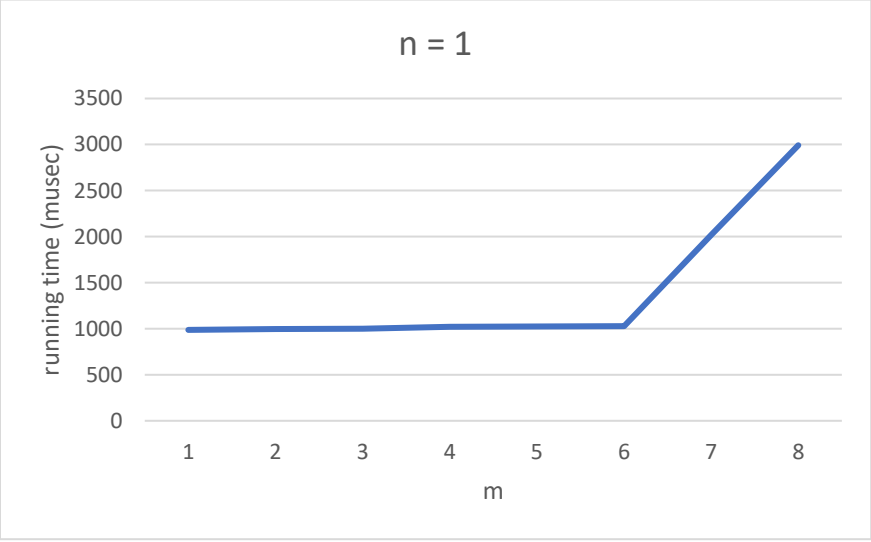
5) Easy Test alloc (1)

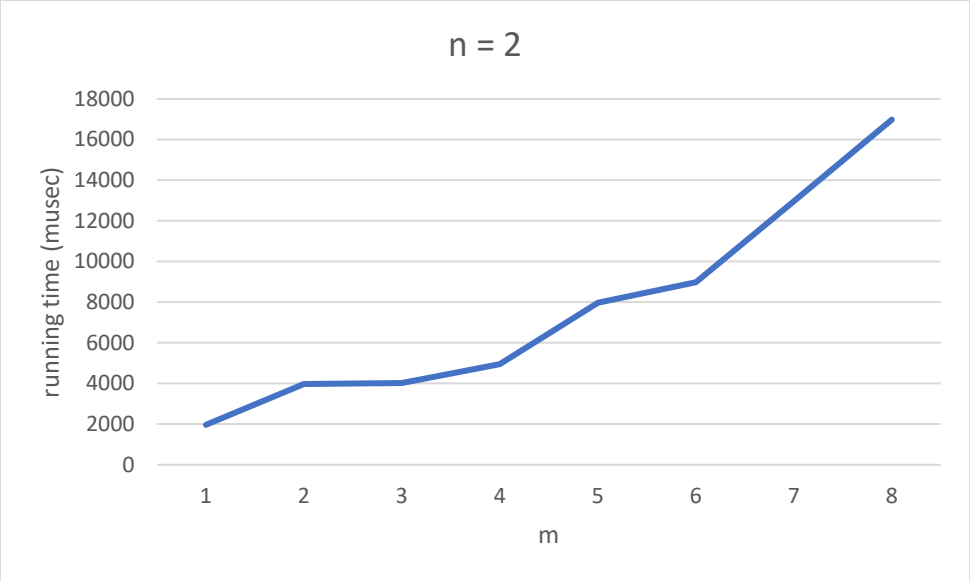
```
PS C:\Users\SuperSteven\Desktop\CSCE 313\Lecture\PA\PA1\Code> ./memtest -b 128 -s 67108864
Printing the Freelist in the format "[index] (block size) : # of blocks"
[0] (128) : 0
Amount of available free memory: 0 bytes
[1] (256) : 0
Amount of available free memory: 0 bytes
[2] (512) : 0
Amount of available free memory: 0 bytes
[3] (1024) : 0
Amount of available free memory: 0 bytes
[4] (2048) : 0
Amount of available free memory: 0 bytes
[5] (4096) : 0
Amount of available free memory: 0 bytes
[6] (8192) : 0
Amount of available free memory: 0 bytes
[7] (16384) : 0
Amount of available free memory: 0 bytes
[8] (32768) : 0
Amount of available free memory: 0 bytes
[9] (65536) : 0
Amount of available free memory: 0 bytes
[10] (131072) : 0
Amount of available free memory: 0 bytes
[11] (262144) : 0
Amount of available free memory: 0 bytes
[12] (524288) : 0
Amount of available free memory: 0 bytes
[13] (1048576) : 0
Amount of available free memory: 0 bytes
[14] (2097152) : 0
Amount of available free memory: 0 bytes
[15] (4194304) : 0
Amount of available free memory: 0 bytes
[16] (8388608) : 0
Amount of available free memory: 0 bytes
[17] (16777216) : 0
Amount of available free memory: 0 bytes
```

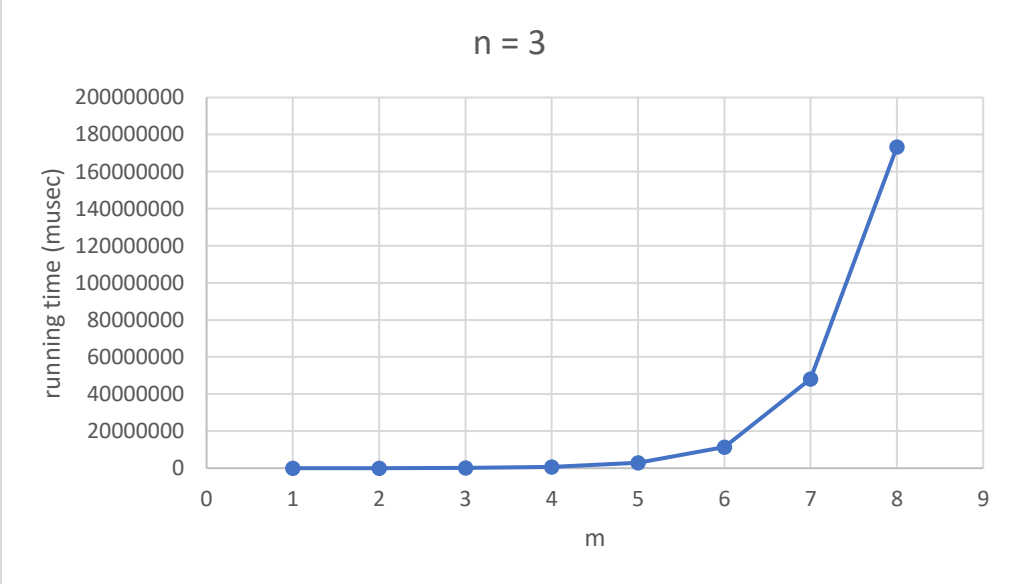
```
[18] (33554432) : 0
Amount of available free memory: 0 bytes
[19] (67108864) : 1
Amount of available free memory: 67108864 bytes
Printing the Freelist in the format "[index] (block size) : # of blocks"
[0] (128) : 1
Amount of available free memory: 128 bytes
[1] (256) : 1
Amount of available free memory: 384 bytes
[2] (512) : 1
Amount of available free memory: 896 bytes
[3] (1024) : 1
Amount of available free memory: 1920 bytes
[4] (2048) : 1
Amount of available free memory: 3968 bytes
[5] (4096) : 1
Amount of available free memory: 8064 bytes
[6] (8192) : 1
Amount of available free memory: 16256 bytes
[7] (16384) : 1
Amount of available free memory: 32640 bytes
[8] (32768) : 1
Amount of available free memory: 65408 bytes
[9] (65536) : 1
Amount of available free memory: 130944 bytes
[10] (131072) : 1
Amount of available free memory: 262016 bytes
[11] (262144) : 1
Amount of available free memory: 524160 bytes
[12] (524288) : 1
Amount of available free memory: 1048448 bytes
[13] (1048576) : 1
Amount of available free memory: 2097024 bytes
[14] (2097152) : 1
Amount of available free memory: 4194176 bytes
[15] (4194304) : 1
Amount of available free memory: 8388480 bytes
[16] (8388608) : 1
Amount of available free memory: 16777088 bytes
[17] (16777216) : 1
Amount of available free memory: 33554304 bytes
[18] (33554432) : 1
Amount of available free memory: 67108736 bytes
[19] (67108864) : 0
Amount of available free memory: 67108736 bytes
Printing the Freelist in the format "[index] (block size) : # of blocks"
[0] (128) : 0
Amount of available free memory: 0 bytes
[1] (256) : 0
Amount of available free memory: 0 bytes
[2] (512) : 0
Amount of available free memory: 0 bytes
[3] (1024) : 0
Amount of available free memory: 0 bytes
```

```
[4] (2048) : 0
Amount of available free memory: 0 bytes
[5] (4096) : 0
Amount of available free memory: 0 bytes
[6] (8192) : 0
Amount of available free memory: 0 bytes
[7] (16384) : 0
Amount of available free memory: 0 bytes
[8] (32768) : 0
Amount of available free memory: 0 bytes
[9] (65536) : 0
Amount of available free memory: 0 bytes
[10] (131072) : 0
Amount of available free memory: 0 bytes
[11] (262144) : 0
Amount of available free memory: 0 bytes
[12] (524288) : 0
Amount of available free memory: 0 bytes
[13] (1048576) : 0
Amount of available free memory: 0 bytes
[14] (2097152) : 0
Amount of available free memory: 0 bytes
[15] (4194304) : 0
Amount of available free memory: 0 bytes
[16] (8388608) : 0
Amount of available free memory: 0 bytes
[17] (16777216) : 0
Amount of available free memory: 0 bytes
[18] (33554432) : 0
Amount of available free memory: 0 bytes
[19] (67108864) : 1
Amount of available free memory: 67108864 bytes
```

6) Timing Graphs







Conclusion:

To sum up, the program works as expected by comparing my result above to Ackerman values to the table. Ackerman functions such as ack (1, x), ack (2, x) and ack (3, x) and easy test with alloc(1) have all been tested with different block size and total memory length (i.e., less than what is needed). The program will not crash or appear any segmentation fault in the case

when there is not enough memory. Besides, in the Main C++ file, getopt() function is used for getting command input such as -b and -s and works perfectly.

As a result of testing Ackerman function, when n is greater than 2 and increasing m from 1 to 8, the value of Ackerman function, the running time of the function and number of allocate/free cycles increase as well. From the data “time taken” increasing from 11 seconds to 173 seconds, we can tell that the tendency of increase in running time is more dramatical when m increases from 6 to 8, and so is the number of allocate/free cycles. Therefore, it’s safe to say that the number of allocate/free cycles is proportional to the running time of Ackerman function. Also, from the graphs above when n is fixed, we can determine the relationship between running time and m which is an exponential growth.

Finally, here is the bottleneck of my program. There is waste of memory when the program tries to determine what the block size for splitting or coalescing a block. To be specific, when my program is trying to find buddies, it explicitly stores the size of the block at the beginning of the allocated block as a part of the header. This wastes memory as the space where the header is being stored cannot be used by the requesting program to store any data.