# Final Proficiency Exam – New Beginnings Summer 2018

## Memory Allocator Project

For this test, you will implement a very basic memory allocator that tracks the memory usage for an Operating System.

The memory allocator will consist of two linked lists:

1. Free Memory List
2. Reserved Memory List

Both are lists of the same type of data – a memory block. A memory block is defined by two integers: the starting block and the block size.

There are a just a few functions that the OS needs from the allocator:

1. Reserve memory
   1. Remove memory from the free list
   2. Add memory to the reserved list
2. Free memory
   1. Remove memory from the reserved list
   2. Add memory to the free list
3. Clean/Merge free memory
   1. Find any blocks in the free list that are adjacent and combine them into a single block

This exam has four(4) levels – 0 through 3 – which add progressive more complicated functionality.

## Getting Started

To get started, fork the following repo from GitHub into your own account:

<https://github.com/jasongraalum/NB2018_Final_ProfTest>

(Don’t clone it – Fork it.)

Then clone your Fork of the repo

%git clone [https://github.com/<your\_acct\_name>/NB2018\_Final\_ProfTest](https://github.com/%3cyour_acct_name%3e/NB2018_Final_ProfTest)

The repo contains:

1. This README.md file(also found on the github repo page.)
2. memtest.cpp – which contains the main function and a series of basic tests for each level. Feel free to add some of your own tests to this file.
3. MemList.hpp – MemList Class Interface file. You can add to this file, but DO NOT CHANGE the existing member functions or data members
4. MemList.cpp – Skeleton of the Class Implementation with the Level 0 implementation completed.
5. MemBlock.hpp and MemBlock.cpp – MemBlock Class Interface and Implementation. It is unlikely that you need to change this file. But, if you do, again, DO NOT CHANGE the existing member functions or data members.
6. A Makefile used to compile and run tests. You can modify this is you need to add any files(however, I wouldn’t expect you will need to do this.)

The main function is provided and includes a series of tests for each level. A Makefile is also provided with targets for running tests of each of the levels. The details for each level are given below.

## Level 0

To get you started, you are provided with the Level 0 member function implementations. Running “make level0” will run the Level0 tests. They should all pass without any implementation updated.

## Level 1

This is the first level were you need to contribute code. The first step is creating a complete constructor for MemList. You also need to add functionality to “move” memory into the Reserve List from the Free List.

Implement the following methods. The prototypes can be found in MemList.hpp. Also, the function skeletons are included in MemList.cpp.:

1. MemList::MemList(unsigned int, unsigned int)
2. MemList::reserveMemBlock(unsigned int)
3. MemList::reservedSize()
4. MemList::freeSize()

Details of the implementation requirements are include in the MemList.hpp and MemList.cpp files.

Run the Level 1 tests to check your implementations

%make level1

Criteria to pass tests:

1. Free and Reserve List sizes are correct after construction
2. Free and Reserve List sizes are correct after reserveMemBlock is called once
3. Free and Reserve List sizes are correct after reserveMemBlock is called multiple times
4. Free and Reserve List sizes are correct after reserveMemBlock even if there isn’t enough memory in the Free List to satisfy the request

## Level 2

The second level extends the implementation to the Free List

Implement the following methods. The prototypes can be found in MemList.hpp. Also, the function skeletons are included in MemList.cpp.:

1. MemList::freeMemBlock(MemBlock \*)
2. MemList::maxFree()
3. MemList::minFree()
4. MemList::freeBlockCount()

Details of the implementation requirements are include in the MemList.hpp and MemList.cpp files.

Run the Level 2 tests to check your implementations

%make level2

Criteria to pass tests:

1. Free and Reserve List sizes are correct after freeMemBlock() is called once
2. Free and Reserve List sizes are correct after freeMemBlock() is called multiple times
3. maxFree() returns the largest block of free memory
4. minFree() returns the smallest block of free memory
5. freeBlockCount() returns the correct number of blocks in the Free List

## Level 3

The third level adds more advanced functionality.

Implement the following methods. The prototypes can be found in MemList.hpp. Also, the function skeletons are included in MemList.cpp.:

1. MemList::defragFree()
2. MemList::reserveMinMemBlock()

Details of the implementation requirements are include in the MemList.hpp and MemList.cpp files.

Run the Level 3 tests to check your implementations

%make level3