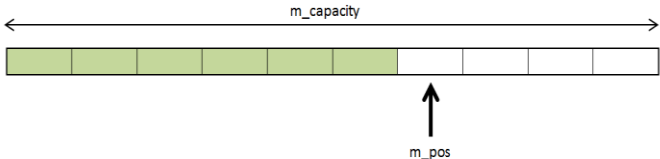


# CSC252 Final

Maximum time allowed <do a good job> (60pts)

Write the class MemBuf that represents memory in a program in the following manner:

```
namespace CSC252Final
{
    class MemBuf
    {
    public:
        MemBuf( ... );
        virtual ~MemBuf(void);
    private:
        char* m_data = nullptr;
        size_t capacity = 0;
        size_t m_pos = 0;
    };
}
```



The MemBuf class has three private data members

- ☐ a char\* pointer named m\_data.
- ☐ a size\_t variable named m\_capacity.
- ☐ a size\_t variable named m\_pos which indicates the position in the char array up to which the MemBuf is used.

Write a main function that shows the use of every one of the functions you write below. [15]

Write code for the following:

- (i) Write a public implementation of an **argument based constructor** for the class MemBuf that takes a size\_t parameter x and allocates the private internal data member m\_data to be large to hold x chars using operator new. You must then fill the character array m\_data with the character '0' completely using a for loop. (do not use bzero) [5].
- (ii) Write a public implementation of a **default constructor** (no args) of the class MemBuf which allocates a default buffer of size 1024 bytes. [5]
- (iii) Write the destructor for class MemBuf. [3]
- (iv) Write a public deep **copy constructor** for class MemBuf. [5]
- (v) Write a public deep copy assignment operator (=) for class MemBuf. [5]
- (vi) Write a public getter method to get the variable m\_capacity of MemBuf. [1]
- (vii) Write a public method called size() which returns the m\_pos variable. [1]
- (viii) Write a public method alloc on the class which takes a size\_t variable len. If m\_pos + len exceeds m\_capacity the method should throw an exception of type std::bad\_alloc. If m\_pos + len is equal to or less than m\_capacity it should advance m\_pos by len and fill the array m\_data with the character 'a' up to the new position to indicate 'allocated'. [5]
- (ix) Write a public method dealloc on the class which takes a size\_t variable len and moves the position variable m\_pos back by len. If len exceeds the current value of m\_pos the method should throw an exception of type std::runtime\_error. If len is less than or equal m\_pos it should retard m\_pos by len and fill the emptied part of the array m\_data with the character '0' to indicate that it is now unallocated. [5]
- (x) Write a global operator<< which prints the contents of the buffer for each location as the 8 bit numeric ascii value of each char. It should do this for only that part of the buffer which is occupied. [5]
- (xi) Write a type conversion operator that returns the m\_data data member as a const value.[5]