Simulator of MMU + paging

Implement in C++:

- Memory manager: a class that manages 32K of RAM in a virtual address space of 64K (16 bits). The missing 32K are to be stored in a swap file (sized for a maximum of 10 processes): 32K*10 on the file.
- On request the memory manager will:
    - Allocate a new page in RAM, returns a virtual address in the 64K address space (to the caller). This is a *unsigned char ** to the VA in 64K;
    - Deallocate a page (identified by the VA pointer);
    - Maintain the page table into the MMU (to start with, simulate the MMU with enough memory to maintain the entire page table in its internal memory);
    - Implement a virtual to physical address space conversion (the physical is 32K);
    - Implement a deref() method that returns a real address in the real Linux address space (this is for practical reasons so that the memory can be accessed for real); so, convert from VA to PA (simulation) then deref() converts from PA to Linux address (valid pointer);
    - Uses the NRU algorithm to evict pages from memory (and store them to the swap file): i.e. use R & M bit to classify pages and evict consequently.
- Process manager: a class that simulates the existence of a process that uses the MMU.
- The process manager will:
    - Manage the requests from a test (only one for now) process that requires addresses in more than 32K of memory (so that an eviction algorithm is required); the test process is a Thread (beware of critical sections if any) that does (on average) 1000 memory references every 5ms (make a thread with a period of 5ms and do 1000 memory requests per cycle); pages must be properly managed, read, written as needed (don't lose data!);
    - To make sure the system is consistent: "Store a run of random addresses (>32K) big enough to be sure more than 32K are used (e.g. generate a sequence of addresses in the range of 36K and store them to a file; the requests can be either a read or write)";
    - Replay the sequence and compare if the memory is consistent with the same "array" (of 64K) generated without the use of the simulated MMU; to do the comparison, use a Thread to read both the paged memory and the 64K array (for testing) and compare that the two representations in memory are consistent; run this test every N second (for N some small integer);