CSE 3150 — C++ Essentials — Yufeng Wu — Fall 2023
Programming Assignment 4. Due: 11:59 pm @ 09/29/2023.

In this assignment, you are to implement the page replacement algorithms for managing virtual memory in a modern operating system. In a modern computer, the virtual memory is organized into pages (or block of memory). We use an integer to represent a page. That is, each page has a distinct integer as its ID. The main memory has a limited capacity and can only hold up to K pages. Here, K is the capacity of the main memory. An important problem regarding to virtual memory is the page replacement: when the memory reaches its limit (i.e., having K pages) and a new page (not currently in memory) is to be add, then need to swap out one current page to make room for this new page. Virtual memory is an important subject in computer science. If you haven't learned about it, it is time to get to know it (you can ask ChatGPT about virtual memory...).

There are many different page replacement algorithms. In this programming assignment, you need to implement the following methods and related functions. These methods will be used to implement the Least Recently Used (LRU) algorithm in the provided test code. That is, you will provide library function calls to support the implementation of the LRU algorithm. Briefly, LRU policy is to evict the page that is least recently used to accomondate a new page to be added to the main memory. The following describes what functions you need to implement, which will be invoked by the LRU algorithm implemented in the provided test code.

1. *void ECPagingInit(int capacity);* Initialize the main memory so that the main memory can hold this number of pages.

2. *void ECPagingDump();* This is not going to be used in grading. But it can be useful for your own testing: dump out the current memory status so you know what is going on.

3. *bool ECPagingIsMiss(int pageCurrent);* return true if we access this page and this access will result in a page miss (i.e., this page is not currently in main memory).

4. *bool ECIsPageReplacing(int pageRequest);* Return true if requesting this specific page would lead to page replacement.

5. *void ECPageReplacementOpt(const std::vector¡int¿ &listAllPageRequests, int indexPageRequest);* This is the optimal page replacement algorithm: swap out the page whose next access is **furthest** away from the current time. Note: for testing purpose, if there is a tie, choose the page with the **smallest** number to swap out. Precondition: memory has reached its capacity and the current page is not in the memory. listAllPageRequests: list of all page requests (ordered by request time) /indexPageRequest: the index of the current page that is being accessed (you can assume this page is not currently in memory) After the function returns, memory is updated with one page swapped out and the page listAllPageRequests[indexPageRequest] is placed in the memory

6. *void ECPageAccessLRU(int pageCurrent);* This is part of the Least Recently Used (LRU) algorithm. Access a page that is either already in the memory or the memory is not full yet. Note: we don't need to evict a page from the main memory in this case.
   Precondition: memory has **not** reached its capacity or and the current page **it** in the memory. pageCurrent: current page to access (which is in memory).

7. *void ECPageReplacementLRU(int pageCurrent);* The LRU page replacement algorithm: swap out the page which is least recently used. This is needed when the memory is full and we need to accommodate a new page that is to be added into the main memory.

Precondition: memory has reached its capacity, and the current page is not in the memory.
pageCurrent: current page to access (which is not in memory).
After the function returns, memory is updated with one page swapped out and pageCurrent is placed in the memory

The following lists things to consider when you implement this function.

1. You should define the data structure of the virtual memory as global variables in the source file (.cpp). This is not the best practice; it would be better to use a class but we haven't learned that yet. For now, just use global variables.
2. Use STL set/vector/map when applicable. These STL utilities have useful methods/functions that can make the implementation easier.
3. Read the code given in the test code to understand how the test code works. This can help you understand more deeply what the functions are meant to do.