PART 1

For Part 1 I implemented

- NRU
- FIFO
- SC
- LRU

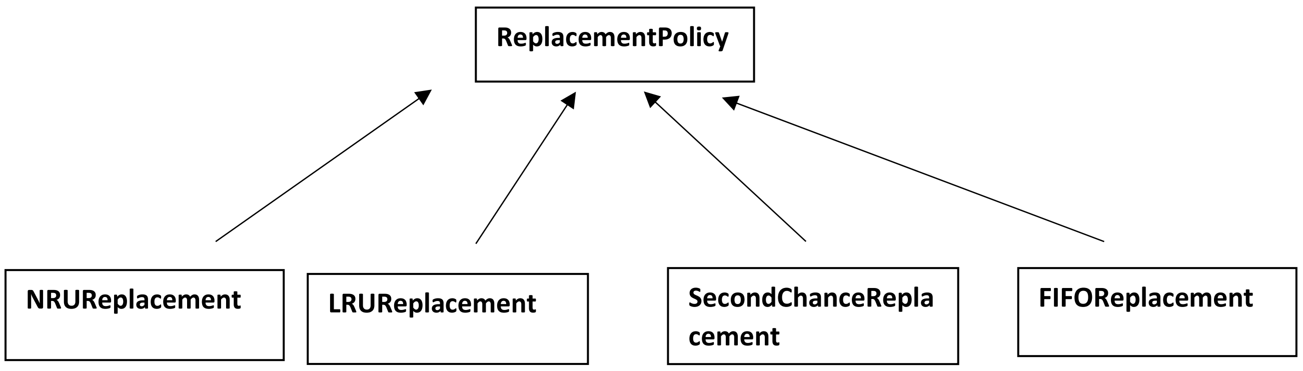
Page replacement algorithm. I couldn't implement WSClock page replacement algorithm.

Firstly, I create **ReplacementPolicy** abstract class.Then I derived for each policy from this abstract class.

This abstract class has two virtual methods.

1. virtual void updatePageTables
2. virtual int getFreePage methods. Then I implemented these methods in derived class.

First method is used to update the paging tables, whenever a memory Access occurs. Since I did not have any interrupt, i have used memory Access and page fault events as an opportunity to update thosse tables. getFreePage method is called whenever OS needs a page replacement. With this implementation Overlaying operating system does not know the details of the replacement algorithm and just uses the proposed page index.



I used P_PageTable struct in updatePageTables and getFreePage methods. This P_PageTable struct is in PageDefinitions.h

```
typedef struct pageTableEntry {
        int frame_Number;
        int present_absent;
        int protection;
        int reference_bit;
        int dirty_bit;
        unsigned int referenced =0;

} PageTableEntry;
```

```
typedef struct pageTable {
        int max_frame_count = 0;
        int current_frame_count = 0;
        P_PageTableEntry pages;

        deque<int> *referencePages;
} PageTable
```

In pageTableEntry struct there are frame_Number, present_absent, protection, reference_bit, dirty_bit, referenced. Present absent variable holds whether the page is in pyhsical memory or not. I did not use protection variable. Dirty bit holds whether the pages modified or not. I used referenced variable in LRU and NRU for choosing the pages.

In pageTable struct there is deque for storing pages. First I thought about use queue in order to ease of implementation of FIFO to choose the pages correct order. Then I realized that I couldn't Access with [] operator to pages then I use deque which has this property. And also in this struct there are max_frame_count and current_frame_count.

Max_frame_count holds how many frames could be in physical memory.

current_frame_count holds the total number of frames.

And also, there is enum in Line 13 in PageDefinitions to count these variables

Implementation Details

Memory

I have implemented the assignment in a layerd approach. At the bottom of the structure lies VirtualMemory class which imitates a mixture part of memeory management, MMU and memory.

Processes:

I have implemented Process using an abstraction called "AbstractProcess" Abstract Process class has a base and limit registers also it has a virtual method call sort. All the processes even (Test and init) is derived from this class. **I did not implemenetd IndexSort Algorithm.**

Operating System:

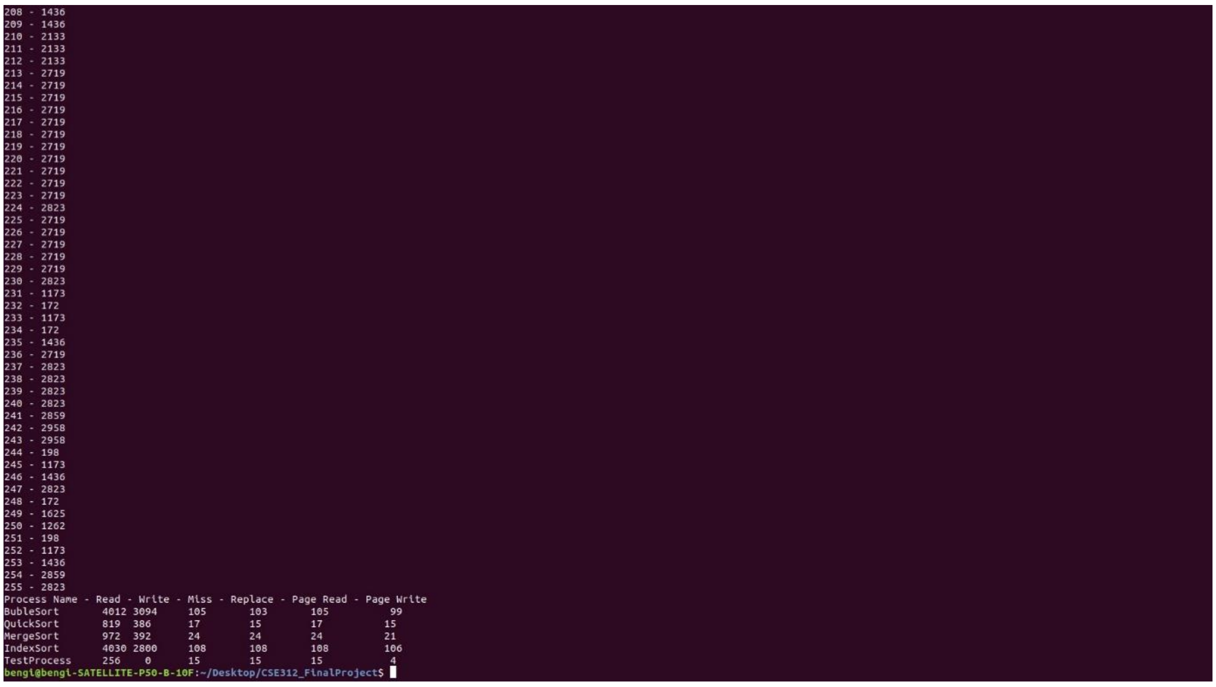
Operating system class is an abstraction of a partial operating system. It provides memory access abstraction with get and set methods (OpSys.cpp line:49, line:66), **synchronizes memory access using semaphores.** Finally, it handles page faults (onPageFaultMethod) and holds page access statistics for each process.

Shortcomings and Failures

1. I did not implement WorkingSet algorithm
2. I did not implement Index sort algorithm.
3. Thread Synchronization is a huge problem whenever I try to use NRU or LRU. However, These algorithms work perfectly, When not using threads. This is probably due to the deque class I used is not thread-safe and I did not realized this error until very late.
4. Finally, I do not use a file base back end, because testing was too long whenever I used a file system.(Since I am using a machine as well.) My virtual memory back-end is another array. But the logic is solid and replacements occur if needed without any problem.

Running Instructions

Just type make and type ./sortArrays 4 2 4 FIFO local 100 abc.dat



```
208 - 1436
209 - 1436
210 - 2133
211 - 2133
212 - 2133
213 - 2719
214 - 2719
215 - 2719
216 - 2719
217 - 2719
218 - 2719
219 - 2719
220 - 2719
221 - 2719
222 - 2719
223 - 2719
224 - 2823
225 - 2719
226 - 2719
227 - 2719
228 - 2719
229 - 2719
230 - 2823
231 - 1173
232 - 172
233 - 1173
234 - 172
235 - 1436
236 - 2719
237 - 2823
238 - 2823
239 - 2823
240 - 2823
241 - 2859
242 - 2958
243 - 2958
244 - 198
245 - 1173
246 - 1436
247 - 2823
248 - 172
249 - 1625
250 - 1262
251 - 198
252 - 1173
253 - 1436
254 - 2859
255 - 2823
Process Name - Read - Write - Miss - Replace - Page Read - Page Write
BubleSort       4012 3094     105       103       105          99
QuickSort       819  386      17        15        17          15
MergeSort       972  392      24        24        24          21
IndexSort       4030 2800     108       108       108         106
TestProcess     256    0      15        15        15           4
bengi@bengi-SATELLITE-P50-B-10F:~/Desktop/CSE312_FinalProject$ 
```