COEN 346: Programming Assignment #3

The main objective of this assignment is to simulate operating system's virtual memory management and concurrency control.

The following issues should be considered for simulation:

1- Processes should be simulated by threads in the application. Each process has a starting time and total duration. "processes.txt" is the input file which contains the number of processes N followed by N lines each contains the process start time and duration.

2- Virtual memory: It consists of a main memory and a large disk space. Both are divided into 'pages'. The size of the main memory will be given as an input file called "memconfig.txt". This file contains a number indicating number of pages in main memory. There is no limitation on the size of disk space. Notice that pages in the main memory should be simulated by an array in the actual computer physical memory, while disk pages are simulated by an array located into a text file such as "vm.txt", which must be accessed every time we need to access the disk.

3- Processes try to store, release and retrieve "variables" from/to the memory. Each page contains only one variable Id and its value. Of course, working with memory can only happen when the process is running. Each variable in the memory has a Last Access property which is a time stamp indicating the last access time to this variable.

4- Virtual memory manager should be a separate thread that receive the commands from other processors and execute them.

5- Virtual memory manager provide three APIs to the processors allowing them to work with the virtual memory. The APIs are as follows:

   a. **Store (string variableId, unsigned int value)**: This instruction stores the given variable id and its value in the first unassigned spot in the memory.

   b. **Release (string variableId)**: This instruction removes the variable id and its value from the memory so the page which was holding this variable becomes available for further storage.

   c. **Lookup (string variableId)**: This instruction checks if the given variableId is stored in the memory and returns back its value or -1 if it does not exist. If the Id exists in the main memory it returns its value. If the Id is not in the main memory but exists in disk space (i.e. page fault occurs), then it should move this variable into the memory and release the assigned page in the virtual memory. Notice that if no spot is available in the memory, program needs to swap this variable with the least recently accessed variable, i.e. the variable with smallest Last Access time, in the main memory.

6- Any process should continually pick one command from a given list of commands located in input file called "commands.txt". The processes then should call the suitable API of virtual memory manager based on the picked command. The process should wait for a random time (from 1 to 1000 milliseconds) between each API call.

   a. **Store** [variable ID] [value]: e.g. "Store 10 5"

   b. **Release** [variable ID]: "Release 10"

c. **Lookup** [variable ID]: “Lookup 10”

7- Output of the program should be a text file “output.txt” which includes the following events:

a. Threads start and finish.

b. Each instruction executed by each thread

c. Page swapping between the main memory and the disk space

Programming Tips:

- In order to protect critical sections and ensure mutual exclusion use appropriate tools, semaphores for instance, within the body of each function.
- The program must work with arbitrary number of threads and arbitrary size of memory.

Following are the list of sample input / output files.

Input Files:

- processes.txt

```
3
2000  3000
1000  2000
4000  3000
```

- memconfig.txt

```
2
```

- “commands.txt”

```
Store 1 5
Store 2 3
Store 3 7
Lookup 3
Lookup 2
Release 1
Store 1 8
Lookup 1
```

Output File:

```
Clock: 1000, Process 2: Started.
Clock: 1010, Process 2, Store: Variable 1, Value: 5
Clock: 1730, Process 2, Store: Variable 2, Value: 3
Clock: 2000, Process 1: Started.
Clock: 2010, Process 1, Store: Variable 3, Value: 7
Clock: 3000, Process 2: Finished.
Clock: 3020, Memory Manager, SWAP: Variable 3 with Variable 1
Clock: 3030, Process 1, Lookup: Variable 3, Value: 7
Clock: 3100, Process 1, Lookup: Variable 2, Value: 3
Clock: 3800, Process 1, Release: Variable 1
Clock: 4000, Process 3: Started.
```

```
Clock: 4200, Process 3, Store: Variable 1, Value 8
Clock: 4400, Memory Manager, SWAP: Variable 1 with Variable 3
Clock: 4410, Process 1, Lookup: Variable 1, Value 8
Clock: 5000, Process 1: Finished.
Clock: 7000, Process 3: Finished.
```

The assignment should be done in a group of two students. The deliverable consists of a well-commented code and a report. Do not include your code in your report.
The report should be at least two pages with the following sections:

- Name of group members
- High level description of the code (description of the methods/functions/threads/data structures and the flow of the program).
- A detailed conclusion, discussing you experience with concurrency control in simulating virtual memory management. You also need to analyze your program and discuss how you enable mutual exclusion in your program and the efficiency of your technique.

This assignment will take 40% of your total mark for programming assignments. Also for this assignment 80% of the mark is dedicated to your code and 20% to your report.

The code and the report should be submitted to the EAS website (https://fis.encs.concordia.ca/eas/).
The deliverable should be submitted through EAS by Sunday April 7th, 11 pm. Demonstrations will be held in the labs during week of April 08.