UTD
CS 4348 Operating Systems
Simulated Memory Management System
Due Date: Sunday, July 27, 2025
100 Points

Overview:
In this project, you will work in groups (each group of two students) to design and implement a simulated memory management system using either Java or C/C++.

You must compile and run this project on UTD Linux machines

Project Objective:
The goal is to develop a simulated memory management system that supports the following features:

- Paging
- Memory allocation and deallocation
- Page replacement algorithms (FIFO and LRU)
- Virtual-to-physical address translation

Project Components
1. Physical Memory Simulation
   - Simulate a fixed-size physical memory (e.g., 1MB divided into 256 frames of 4KB each).
   - Each frame should be capable of storing one page.
   - Maintain a frame table to track free and occupied frames.

2. Virtual Address Space
   - Simulate multiple processes, each with its own page table.
   - Processes can request memory allocation in units of pages.

3. Page Table
   - Each process maintains its own page table.
   - Each Page Table Entry (PTE) should include:
     - Valid bit
     - Frame number
     - Reference bit (used for LRU)
     - Modified bit

4. Memory Allocation
   - Support the following user command:
     alloc <process_id> <num_pages>
     This command allocates the requested number of pages to the specified process and updates the page table accordingly.

5. Memory Access

- Support the following user command:
  access <process_id> <virtual_address> <read/write>
  This command translates a virtual address to a physical address using the process's page table.
  If the page is not in memory (page fault), a page replacement algorithm is triggered.

6. Page Replacement Algorithms

- Implement the following two algorithms for handling page faults:
  - FIFO (First-In-First-Out)
  - LRU (Least Recently Used)

7. Memory Deallocation

- Support the following user command:
  free <process_id>
  This command deallocates all memory used by the specified process and removes its page table.

==========

Sample Input:
alloc P1 5
alloc P2 3
access P1 8192 read
access P1 12288 write
access P2 4096 read
free P1

Sample Output:
Allocated 5 pages to process P1
Allocated 3 pages to process P2
Translated virtual address 8192 (P1) → physical address 24576
Translated virtual address 12288 (P1) → physical address 28672
Translated virtual address 4096 (P2) → physical address 16384
Freed memory for process P1

What to submit:

- Source code with documentation
- How to run it
- A sample input/output file showing memory operations