OS (CS3523) LAB EXAM : PROBLEM 2

ABHAY SHANKAR K: CS21BTECH11001

1. FIFO

(1) All code within `FIFO()`.
(2) Opens file in read-only mode and reads the page size and frame number.
(3) Then initialises a `deque<int>`, which is how we implement FIFO.
(4) Tries to read an address from file. If EOF is encountered, returns.
(5) Finds the page number of the address and stores in `acc`.
(6) Checks if the page is already present - if it is, repeat 4.
(7) If the page is not in the deque, a fault occurs. If there is no empty frame, we dequeue the appropriate page as dictated by FIFO and then enqueue the faulting page. Otherwise we enqueue directly.
(8) Repeat 4.

2. LRU

(1) All code within `LRU()`.
(2) Opens file in read-only mode and reads the page size and frame number.
(3) Then initialises a `deque<int>`. We implement LRU within this deque.
(4) Tries to read an address from file. If EOF is encountered, returns.
(5) Finds the page number of the address and stores in `acc`.
(6) If the page is already present, moves it to the front of the deque. Repeat 4.
(7) If page faults, we check if a free frame exists. If it does, we push the new page onto the front of the queue. If there is no free frame, we pop and evict the page at the back of the queue. This is the LRU page, as each usage moves a page to the front of the queue. We then push the new page onto the front of the queue.
(8) Repeat 4.
(9) This does not store the frame list at any point in time, it only stores the history of allocation.

3. OPT

(1) All code within `OPT()`.
(2) Opens file in read-only mode and reads the page size and frame number.
(3) Then initialises a `deque<int>`. We load the entire list of pages into this deque.
(4) Pop the first page in the deque.
(5) If the page is already present in memory, repeat 4.
(6) If page faults, we check if a free frame exists. If it does, we push the new page onto the back of the queue. If there is no free frame, we select the frame containing the appropriate page as dictated by OPT, i.e. the one used farthest in the future, and replaces it with the faulting page.

(7) Repeat 4.

4. Summary

- Due to alignment errors (the program treated the frame size and number as addresses), the file is reopened and closed in each paging function.
- The program takes a very long time to run if the page size exceeds 10^9. This may be due to excessive IO. Please do not enter larger values :).
- Naturally, the file is closed at the end of each paging function.