Department of Computer Science & Engineering

Course Title	: Operating System Lab
Course Code	: CSE 406
Experiment Name	: Least Recently Used(LRU) Page Replacement Algorithm.
Submission Date	: 01.05.25

Submitted To:	**Submitted By:**
Atia Rahman Orthi	**Susmita Roy**
Lecturer,	**Reg: 21201199**
Department of CSE, UAP	**Sec: B2**

Problem Statement:

Implement the Least Recently Used(LRU) Page Replacement Algorithm.

pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3]

frame_size = 4

Solving Process Steps:

Step 1: Initialization

- pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3]: List of page references.
- frame_size = 4: Number of available memory frames.
- frames = []: Current pages in memory.
- page_faults = 0: Counter to track page faults.

Step 2: Iterate Over Page References

For each page in pages:

- If page is NOT in frames (page fault):
- Increment page_faults.
- If there's still space in frames, append the page.
- If frames is full:
- Remove the least recently used page (the one at index 0).
- Append the current page to the end.
- If page IS in frames (page hit):
- Remove it from its current position (to update its recent use).
- Append it again (making it the most recently used).

Step 3: Final Output

- Print the total number of page faults.

Source Code:

```python
pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3]
frame_size = 4

frames = []
page_faults = 0

for page in pages:
    if page not in frames:
        page_faults += 1
        if len(frames) < frame_size:
            frames.append(page)
        else:
            frames.pop(0)
            frames.append(page)
    else:
        frames.remove(page)
        frames.append(page)

print(f"\nTotal Page Faults: {page_faults}")
```

Output:

```
Total Page Faults: 6
```

Discussion:

Advantages:

- Better Performance: LRU reduces unnecessary page replacements by keeping recently used pages in memory.
- Realistic: Reflects actual usage patterns in most programs.
- Widely Used: Many real-world OS systems implement or approximate LRU behavior.

Disadvantages:

- Costly Operations: Maintaining access order (e.g., with list operations) can be inefficient.
- More Complex: Compared to FIFO, LRU requires additional logic to reorder frames.
- Scalability Issue: As the number of pages grows, the time to search and reorder also increases without optimization.

Conclusion:

This code uses the LRU page replacement method, which removes the page that hasn’t been used for the longest time when a new page needs to be loaded. It works better than FIFO because it keeps recently used pages in memory. Although it's a bit more complicated and may slow down with many pages, it's still a good and commonly used method in real systems.