Lab Assignment 8

Name: Manobal Singh Bagady
SID: 21104129
Semester: 8
Branch: Electrical Engineering
Due Date: Apr 14, 2025

Code:

```
from collections import deque
from prettytable import PrettyTable


class Process:
    def __init__(self, pid, mem_req, page_size):
        self.pid = pid
        self.mem_req = mem_req
        self.page_size = page_size
        self.num_pages = (mem_req + page_size - 1) // page_size
        self.page_table = {i: None for i in range(self.num_pages)}



def simulate_paging(total_mem, frame_size, processes):
    total_frames = total_mem // frame_size
    free_frames = deque(range(total_frames))
    loaded_pages = deque()  # FIFO queue of (pid, page)

    for proc in processes:
        print(
            f"\n--- Allocating Process P{proc.pid} ({proc.mem_req}
bytes, {proc.num_pages} pages) ---"
        )
        for page in range(proc.num_pages):
            if free_frames:
                frame = free_frames.popleft()
                proc.page_table[page] = frame
                loaded_pages.append((proc.pid, page))
                print(f"  Page {page} → Frame {frame}")
```

```
            else:
                # FIFO replacement
                old_pid, old_page = loaded_pages.popleft()
                victim = next(p for p in processes if p.pid == old_pid)
                victim_frame = victim.page_table[old_page]
                print(
                    f"  [FULL] Evicting P{old_pid}-Pg{old_page} from
Frame {victim_frame}"
                )
                victim.page_table[old_page] = None
                proc.page_table[page] = victim_frame
                loaded_pages.append((proc.pid, page))
                print(f"  Page {page} of P{proc.pid} → Frame
{victim_frame}")

    # Build PrettyTable for each process
    print("\n=== Final Page Tables ===")
    for proc in processes:
        table = PrettyTable()
        table.field_names = ["Page#", "Frame# / Status"]
        for pg, fr in proc.page_table.items():
            status = str(fr) if fr is not None else "Not in Memory"
            table.add_row([pg, status])
        print(f"\nProcess P{proc.pid} Page Table:")
        print(table)

    # Build a global frame allocation table
    frame_alloc = {f: "Free" for f in range(total_frames)}
    for proc in processes:
        for pg, fr in proc.page_table.items():
            if fr is not None:
                frame_alloc[fr] = f"P{proc.pid}-Pg{pg}"

    alloc_table = PrettyTable()
    alloc_table.field_names = ["Frame#", "Owner"]
    for f in range(total_frames):
        alloc_table.add_row([f, frame_alloc[f]])
    print("\n=== Frame Allocation ===")
    print(alloc_table)
```

```
if __name__ == "__main__":
    total_mem = int(input("Enter total physical memory size (bytes): "))
    frame_size = int(input("Enter frame size (bytes): "))
    n = int(input("Number of processes: "))
    processes = []
    for i in range(1, n + 1):
        mem_req = int(input(f" Process {i} memory requirement (bytes): 
"))
        processes.append(Process(pid=i, mem_req=mem_req, 
page_size=frame_size))
    simulate_paging(total_mem, frame_size, processes)
```

Output: