Assignment-3

NAME- Abdul Waheed Al Faaiz

Course- BTech CSE Core -7

Roll No.- 2301010468

Task1.

Implement priority and round robin scheduling algorithms. Calculate turnaround and average waiting time

Code.

```
            if remaining[i] > 0:
                done = False
                if remaining[i] > quantum:
                    t += quantum
                    remaining[i] -= quantum
                else:
                    t += remaining[i]
                    waiting_time[i] = t - processes[i][1]
                    remaining[i] = 0
        if done:
            break

    for i in range(n):
        turnaround_time[i] = processes[i][1] + waiting_time[i]

    print("\nRound Robin Results:")
    print("PID\tBurst\tWaiting\tTurnaround")
    for i in range(n):
        print(f"{processes[i][0]}\t{processes[i][1]}\t{waiting_time[i]}\t{turnaround_time[i]}")


f main():
    n = int(input("Enter number of processes: "))
    processes = []

    for i in range(n):
        pid = int(input(f"Enter PID for process {i+1}: "))
        burst = int(input(f"Enter burst time for process {i+1}: "))
        priority = int(input(f"Enter priority for process {i+1}: "))
        processes.append([pid, burst, priority])

    priority_scheduling(processes)

    quantum = int(input("\nEnter time quantum for Round Robin: "))
    round_robin(processes, quantum)


__name__ == "__main__":
    main()
```

Output.

```
Enter choice: 1

--- Task 1: Process Creation ---
Child PID: 576, Parent PID: 515, Message: Hello from child 0
Child PID: 577, Parent PID: 515, Message: Hello from child 1
Child PID: 578, Parent PID: 515, Message: Hello from child 2

=== OS Lab Assignment Menu ===
1. Process Creation
2. Command Execution using exec()
3. Zombie & Orphan Processes
4. Inspect Process Info from /proc
5. Process Prioritization
6. Exit
```

```
┌──(kalilinux㉿DESKTOP-71JCI24)-[~/OS_Lab_1]
└─$ python3 process_management.py --task 1 --n 3
[TASK1] Parent PID 730 creating 3 children
Parent: created child 731
Parent: created child 732
Child 1: PID=731, PPID=730, msg='Hello from child 1'
Child 2: PID=732, PPID=730, msg='Hello from child 2'
Parent: created child 733
Parent: waiting for children to finish...
Parent: reaped child 731 (status 0)
Parent: reaped child 732 (status 0)
Child 3: PID=733, PPID=730, msg='Hello from child 3'
Parent: reaped child 733 (status 0)
[TASK1] Done.
```

Task2.

Simulate worst fit, best fit and first fit memory allocation strategies

Code.

```
# ------------------ Main Program ------------------
if __name__ == '__main__':
    print("System Starting...")

    # Create processes
    p1 = multiprocessing.Process(target=system_process, args=('Process-
    p2 = multiprocessing.Process(target=system_process, args=('Process-

    # Start processes
    p1.start()
    p2.start()

    # Wait for processes to complete
    p1.join()
    p2.join()

    print("System Shutdown.")
```

Output.

```
Enter choice: 2

--- Task 2: Command Execution Using exec() ---
Child executing 'ls -l'
total 4
-rw-r--r-- 1 kalilinux kalilinux 2873 Oct  1 23:58 process_management.py

=== OS Lab Assignment Menu ===
1. Process Creation
2. Command Execution using exec()
3. Zombie & Orphan Processes
4. Inspect Process Info from /proc
5. Process Prioritization
6. Exit
```

```
┌──(kalilinux㉿DESKTOP-71JCI24)-[~/OS_Lab_1]
└─$ python3 process_management.py --task 2 --n 2 --cmd date --no-exec
[TASK2] Parent PID 737 creating 2 children to run ['date'] (use_exec=Fal
se)
Parent: forked child 738
[child 738] will run: ['date']
Parent: forked child 739
[child 739] will run: ['date']
Thu Oct  2 06:49:18 PM IST 2025
Thu Oct  2 06:49:18 PM IST 2025
Parent: reaped child 738 (status 0)
Parent: reaped child 739 (status 0)
[TASK2] Done.
```

Task3.

Implement mft(fixed partitions) and mvt(variable partitions) strategies.

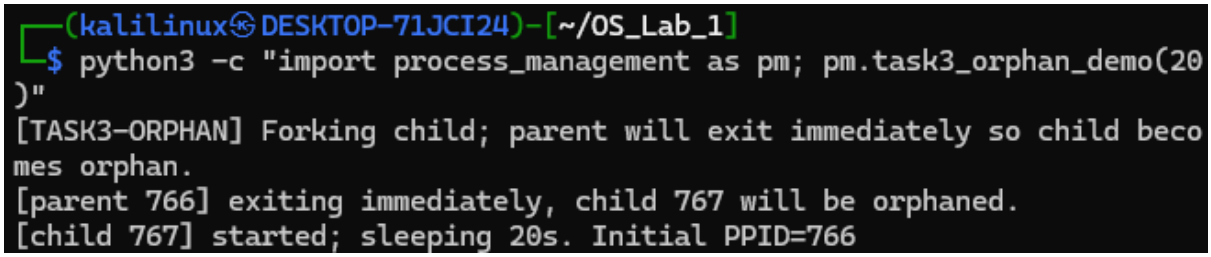
Code.

```
# ------------------- Main Program -------------------
if __name__ == '__main__':
    print("System Starting...")

    # Create processes
    p1 = multiprocessing.Process(target=system_process, args=('Process-1',))
    p2 = multiprocessing.Process(target=system_process, args=('Process-2',))

    # Start processes
    p1.start()
    p2.start()

    # Wait for processes to complete
    p1.join()
    p2.join()

    print("System Shutdown.")
```

```
  GNU nano 8.6                 system_simulation.py
# Import required libraries
import multiprocessing
import time
import logging

# ------------------ Logging Setup ------------------
logging.basicConfig(
    filename='process_log.txt',    # Log file
    level=logging.INFO,            # Logging level
    format='%(asctime)s - %(processName)s - %(message)s'
)

# ------------------ Process Function ------------------
def system_process(task_name):
    """Simulates a system process by logging start and end, with delay.>
    logging.info(f"{task_name} started")
    time.sleep(2)  # Simulate process execution
    logging.info(f"{task_name} ended")

# ------------------ Main Program ------------------
if __name__ == '__main__':
    print("System Starting...")

    # Create processes
    p1 = multiprocessing.Process(target=system_process, args=('Process->
    p2 = multiprocessing.Process(target=system_process, args=('Process->

    # Start processes
    p1.start()
    p2.start()
```

Output.

```
  GNU nano 8.6                    process_log.txt
2025-10-02 20:25:37,257 - Process-1 - Process-1 started
2025-10-02 20:25:37,258 - Process-2 - Process-2 started
2025-10-02 20:25:39,258 - Process-1 - Process-1 ended
2025-10-02 20:25:39,259 - Process-2 - Process-2 ended
```


(kalilinux㉿DESKTOP-71JCI24)-[~/OS_Lab_1]
$ python3 -c "import process_management as pm; pm.task3_orphan_demo(20
)"
[TASK3-ORPHAN] Forking child; parent will exit immediately so child beco
mes orphan.
[parent 766] exiting immediately, child 767 will be orphaned.
[child 767] started; sleeping 20s. Initial PPID=766