SYSC 4001
Assignment 2

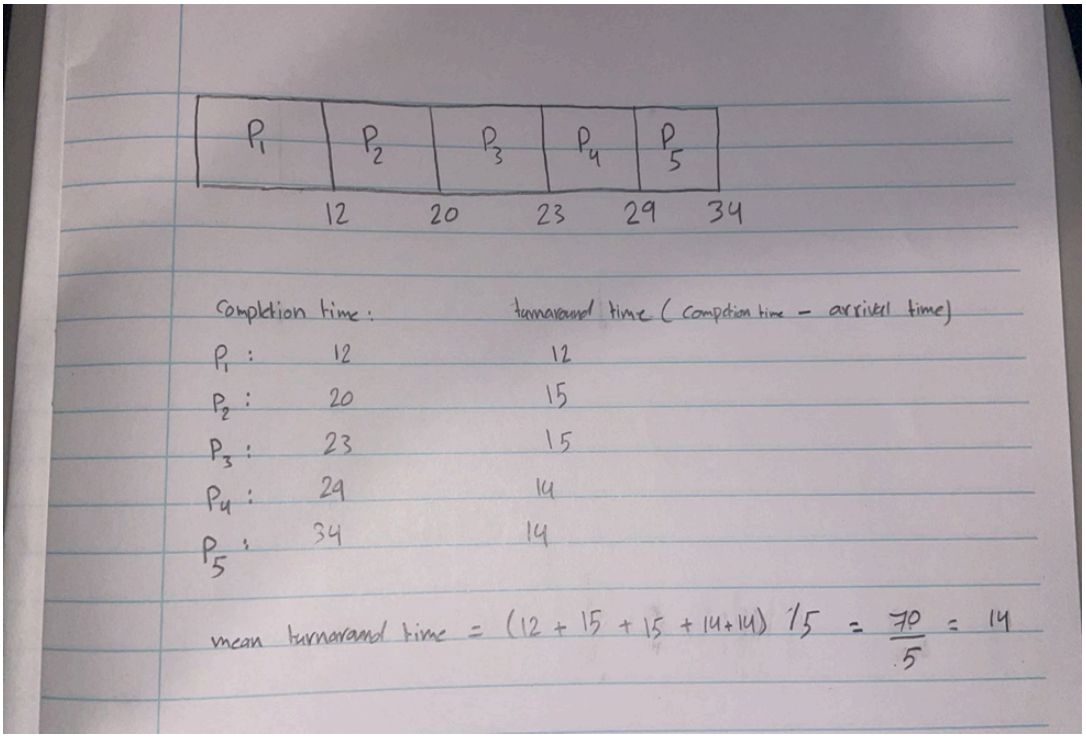
Shahd Elsaman
101268283
Nawal Musameh
101360700

Github link:
https://github.com/ShahdElsaman/SYSC4001_A2_P2.git

https://github.com/ShahdElsaman/SYSC4001_A2_P3.git

Part I – Concepts [1.5 marks] Question a)

(i) The First Come First Serve (FCFS) scheduling algorithm runs processes in the order they arrive[1]. It uses a simple First-In First-Out (FIFO) queue where each process waits its turn, and once a process starts running, it continues until it finishes or performs an I/O operation. FCFS is fair because every process gets the CPU based on when it arrives, but it can cause long waiting times if a long process runs before shorter ones, which is known as the convoy effect[2]. Even though it is not the most efficient for all situations, FCFS is easy to understand, simple to implement.

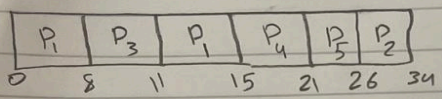


ii) Its scheduling algorithm method is used to manage and schedule multiple processors. Round Robin gives all processors equal quantum time to run regardless of their priorities. The round robin works as follows, when the processors arrive they enter the queue and then each processor will be given an equal CPU time to run. When the first process starts executing (running) it will be given a specific quantum time. If the process finishes the job it's assigned to within the quantum time it gets terminated and exits. However, if it needs more time the process pauses and saves its state then goes back to the queue until it can start running again. The cycle keeps going and it gives every process the same time slice.

	completion time	turnaround
P_1	19	19
P_2	23	18
P_3	15	7
P_4	33	18
P_5	34	14

mean turnaround time is $\frac{19+18+7+18+14}{5} = \frac{76}{5} = 15.2\text{ms}$

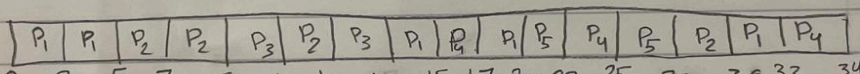
(iii) SJF with preemption



	Completion Time	Turnaround Time
P_1	15	15
P_2	34	29
P_3	11	3
P_4	21	6
P_5	26	6

mean turnaround time = $\frac{15+29+3+6+6}{5}$ = 11.5 ms

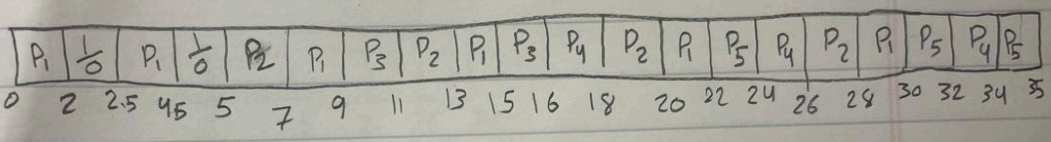
(iv) multiple queues with feedback



	completion time	turnaround time
P_1	33	33
P_2	25	25
P_3	14	6
P_4	34	19
P_5	28	8

mean turnaround time = $\frac{33+25+6+19+8}{5}$ = 18.2 ms

b) FCFS

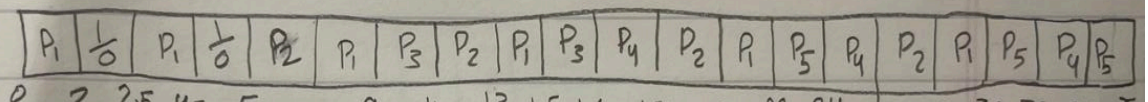


	Completion time	turnaround time
P_1	30	30
P_2	23	23
P_3	16	8
P_4	34	19
P_5	35	15

mean turnaround time $= \frac{30 + 23 + 8 + 19 + 15}{5}$

$= 19$ ms

b) RR (Round-robin)

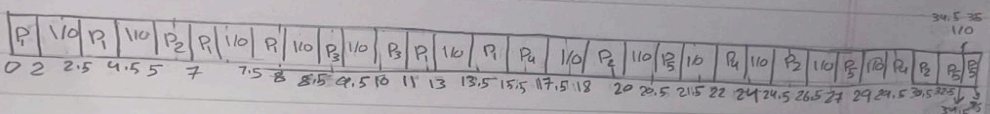


	Completion time	turnaround time
P_1	30	30
P_2	23	23
P_3	16	8
P_4	34	19
P_5	35	15

mean turnaround time = $\frac{30 + 23 + 8 + 19 + 15}{5}$

= 19 ms

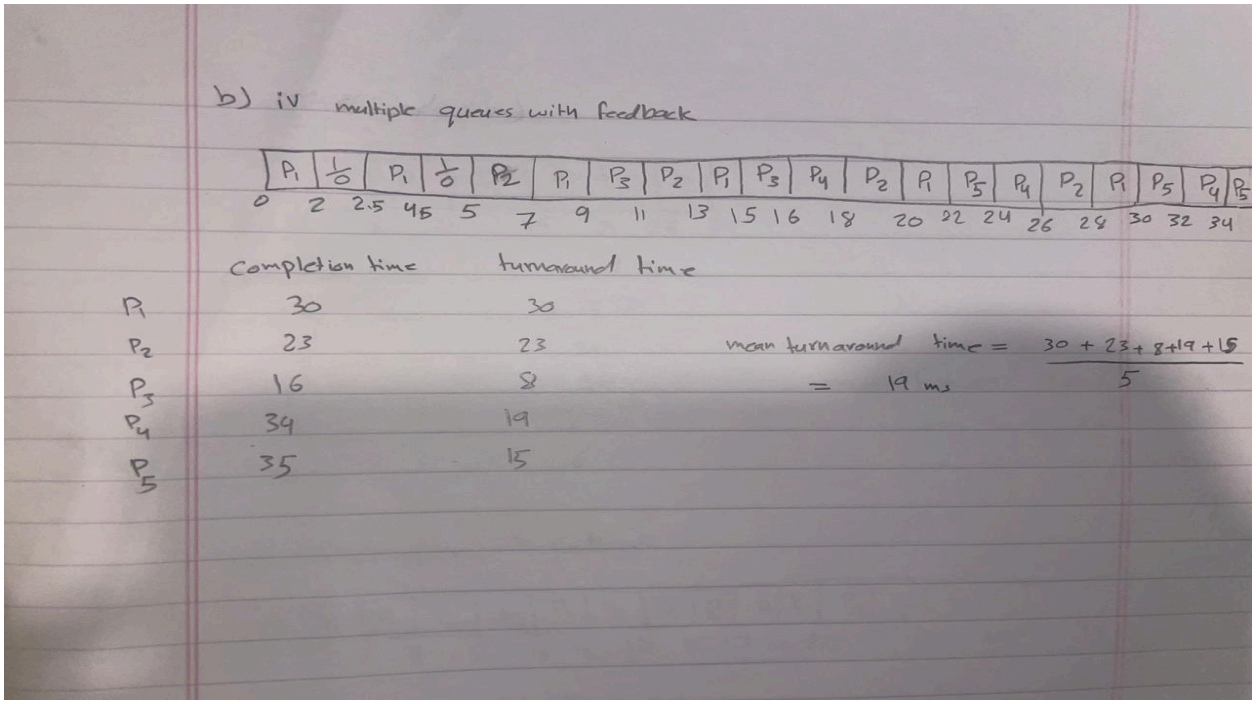
b) iii SJF with preemption (SRTF)



	Completion time	turnaround time
P_1	15.5	15.5
P_2	32.5	27.5
P_3	11	3
P_4	30.5	15.5
P_5	36	16

mean turnaround time = $\frac{15.5 + 27.5 + 3 + 15.5 + 16}{5}$

= 15.5



c)

1) Determine which free partition will be allocated to each job for the following algorithms

(i) First fit

Job	Allocated position	Original size vs New
J1	Position 2	340 - 140 = 200
J2	Position 1	85 - 82 = 3
J3	Position 8	280- 275 = 5
J4	Position 2 (leftover)	200 - 65 = 135
J5	Position 4	195 -190 = 5

Free memory after allocations = 3 + 135 + 28 +5 +55+160+75+5 = 466 KB
Internal fragmentation = 0 KB
External fragmentation = 466 KB

(ii) Best fit

Job	Allocated position	Original size vs New
J1	Position 6	160 - 140 = 20
J2	Position 1	85 - 82 = 3
J3	Position 8	280- 275 = 5
J4	Position 7	75 - 65 = 10
J5	Position 4	195 -190 = 5

Free memory after allocations = 3 + 340 + 28 +5 +55+20+10+5 = 466 KB
Internal fragmentation = 0 KB
External fragmentation = 466 KB

(iii) Worst fit

Job	Allocated position	Original size vs New
J1	Position 2	340 - 140 = 200
J2	Position 8	280 - 82 = 198
J3	N/A	Fails (because the hole >= 275)
J4	Position 2	200 - 65 = 135
J5	Position 4	198 -190 = 8

Free memory after allocations = 85 + 135 + 28 +195 +55+160+75+8 = 741 KB
Internal fragmentation = 0 KB
External fragmentation = 761 KB

2) First Fit: This method is fast, but it can scatter large free blocks early, leaving smaller fragments scattered in memory. In this scenario, it allocated all jobs but left some external fragmentation.

Best Fit: Best Fit tries to minimize wastage of large holes by using the smallest suitable hole, though it can create many small, unusable fragments. Here, it left one large free block, which is better for future large jobs.

Worst Fit: Worst Fit attempts to avoid small fragments by using the largest holes first, but it can quickly break large blocks into medium ones, preventing the allocation of later large jobs, as seen with Job 3.

Part II – Concurrent Processes in Unix [1 mark]

1) The exec function: The exec() system call is used in UNIX-like operating systems to run a new program inside an existing process. When a process calls exec(), the existing program is immediately terminated when exec() is called, the new program is loaded into memory and completely replaces the old one[3], including its code, data, heap, and stack. The process keeps the same process ID (PID), but its memory and execution context are overwritten by the new program, which then starts running from its main entry point. Because of this, exec() does not create a new process, it transforms the current one into a different program. This system call is often used right after a fork(), where a parent process creates a child and the child immediately calls exec() to load and run another program[4], such as when a shell runs user commands. If exec() is called successfully, it never returns, if it fails, it returns an error. In short, exec() is a key mechanism that allows a process to replace itself with a new program while keeping the same identity in the system.
2) Fork is used to create a new process (child). It can be called by the parent to create a child process. The child can share the same resources as the parent however, it has its own memory and PID. Both parent and child run at the same time and the child needs to be terminated first before the parent.

References:

Galvin, Abraham S., Greg Gagne, Peter Baer Galvin, and Jillian P. Galvin. Operating System Concepts. 7th ed. Hoboken, NJ: John Wiley & Sons, [Year of Publication]. PDF file.

---

[1] Operating System Concepts, Seventh Edition, Page 158

[2] Operating System Concepts, Seventh Edition, Page 159

[3] Operating System Concepts, Seventh Edition, Page 51

[4] Operating System Concepts, Seventh Edition, Page 748