20 points) Consider the following set of five processes where arrival is the time the process became ready, t is the total service time, and e is the external priority. Assume that execution starts immediately at time 0 and there is no context switch overhead.

Process	Arrival	*t*	*e*
p0	0	80	9
p1	15	25	10
p2	15	15	9
p3	85	25	10
p4	90	10	11

For the following scheduling disciplines, draw a time diagram showing when each of the five processes is executed. (In the case of a tie, assume that the process with the lower process number executes first.) Calculate the average waiting time for each of the scheduling disciplines.

a) FIFO
b) SJF
c) SRT
d) RR (quantum = 10)
e) ML (using FIFO at each priority level)

a)

FIFO (FCFS)

P0	P1	P2	P3	P4

0 — 80 (+25) — 105 (+15) — 120 — 145 (+10) — 155

Completion Time (CT) = P_0, 80 P_3, 145
P_1, 105 P_4, 155
P_2, 120

Waiting Time = CT - AT

$P_0 = 0$ b/c (starts right away.)

$P_1 = 80 - 15 = 65$

$P_2 = 105 - 15 = 90$

$P_3 = 120 - 85 = 35$

$P_4 = 145 - 90 = 55$

245

Avg WT = 245/5 = 49

b)

(B) SJF

P_0	P_4	P_2	P_1	P_3
0 (+80)	80 (+10)	90 (+15)	105 (+25)	130 (+25) 155

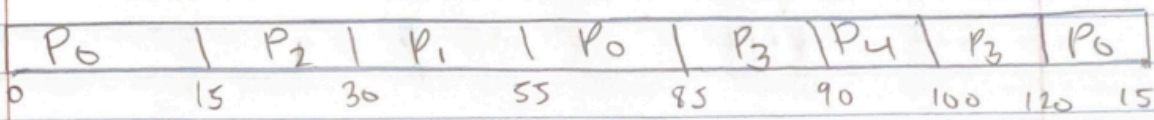
Completion time, $P_0 = 80$, $P_1 = 130$
$P_2 = 105$, $P_3 = 155$
$P_4 = 90$.

Waiting time =
$P_0 = 0$ (Start right away)
$P_1 = 105 - 15 = 90$
$P_2 = 90 - 15 = 75$
$P_3 = 130 - 85 = 45$
$P_4 = 80 - 90 = 0$ *(can't be negative starts on arrival)
210

WT = 210 / 5 = 42

c)

SRT



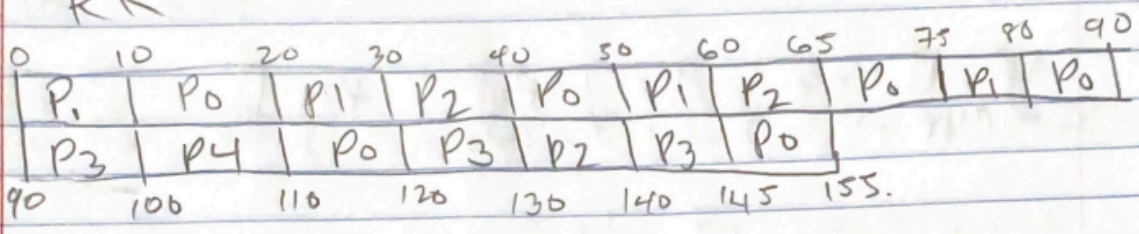
Avg WT =

$$\frac{(0-0) + (15+15) + (30-15) + (55-15) + (25-85) + (90-90) + (100-90) + (120-85)}{5}$$

$$= \frac{100}{5} = \boxed{20}$$

d)

RR



Avg WT Time:

$$\left(0+0+15+15+20+20+10+15+5+5+5+10+20+20+10+10+5\right) = 185/5$$

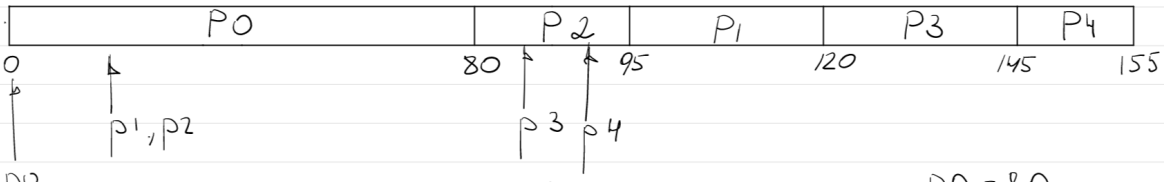
$$= 37$$

e)

e) ML (using FIFO at each priority level)

Priority levels:

1) 9 - p0, p2
2) 10 - p1, p3
3) 11 - p4



Completion Time:

p0 - 80
p2 - 95
p1 - 120
p3 - 145
p4 - 155

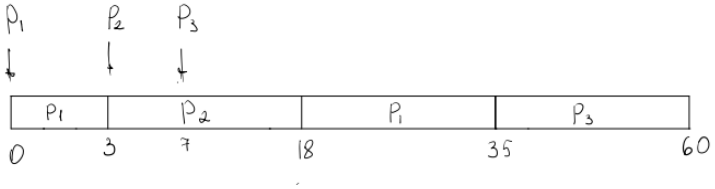
Average Waiting Time =

$$= \frac{0 + (80 - 15) + (95 - 15) + (120 - 85) + (145 - 90)}{5} = \boxed{47}$$

2. (20 Points) Recall that for the Shortest Remaining Time (SRT) scheduling discipline, the priority function P= -(t-a), where (t-a) is the remaining time; (t= total service time, a = attained service time) the decision mode is preemptive; and the arbitration rule is chronological or random among process with the same time to completion. Now, consider a scheduling discipline called Longest Remaining Time (LRT) which has the same decision mode and arbitration rule as SRT, but with a priority function of P=(t-a). Given the set of three processes listed below, which of these two scheduling disciplines will result in less overhead due to context switching? You must explain your answer. (Hint: You may choose to draw timelines or Gantt Charts as part of your answer.)

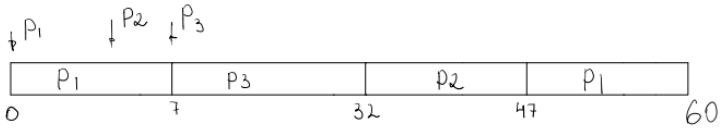
- Process 1 will take 20 cycles and enters the system at time 0.
- Process 2 will take 15 cycles and enters the system at time 3.
- Process 3 will take 25 cycles and enters the system at time 7.

SRT



AWT = $\frac{(18-3)+0+(35-7)}{3}$=14.3

LRT



AWT = $\frac{(47-7)+(32-3)+0}{3}$=23

Even though LRT and SRT have the same number of context switching = 3, the SRT, in this case, will result in much less overhead due to a smaller average waiting time of 14.3 compared to 23 in LRT.

-

3. (10 Points)What are the types of load- balancing available in multiple-processor scheduling? Which load balancing mechanism checks load on each processor to load tasks? Differentiate between soft affinity and hard affinity.

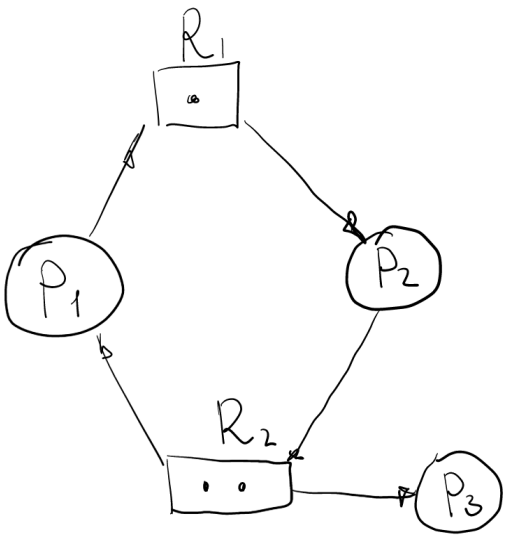
Load balancing: attempts to keep workload evenly distributed
Types of load balancing:

- Push migration - periodic task checks load on each processor, and if found pushes task from overloaded CPU to other CPU's
- Pull migration - idle processor pulls waiting task from busy processor
  The push migration load balancing mechanism checks load on each processor to load tasks.
- Soft affinity - the operating system attempts to keep a thread running on the same processor, but no guarantees.
- Hard affinity - allows a process to specify a set of processes it may run on.

  The soft affinity tries to complete a process in a given deadline, but it doesn't guarantee its completion at that exact time. The hard affinity, however, guarantees the completion at the given deadline and is only assigned to a specific process.

4. (10 Points)Write a short program in C/C++ that creates two processes that deadlock. You MUST remove the deadlocked processes from the system (kill -9) once you have completed your experiments.

5. (20 Points) The question involves a resource allocation graph (RAG) with three processes (P1, P2, P3) and two resources (R1 with 1 instance and R2 with 2 instances). The current system state is: P1 holds one instance of R2 and waits for R1, P2 holds R1 and waits for one instance of R2, and P3 holds one instance of R2. Draw the RAG, determine if there is a deadlock, and propose solutions if a deadlock exists.



P2 holds R1 and waits for R2. Since P3 is only holding one instance of R2 without waiting for any resource, P2 could potentially be preempted from R1, allowing P1 to acquire R1 and thus avoiding a deadlock.

6. (20 Points) Consider the following maximum-claim reusable resource system with four processes (P0, P1, P2, P3) and three resource types (R0, R1, R2). The maximum claim matrix is given by

$$C = \begin{bmatrix} 4 & 1 & 4 \\ 3 & 1 & 4 \\ 5 & 6 & 13 \\ 1 & 1 & 6 \end{bmatrix}$$

where C_{ij} denote maximum claim of process I for resource j. The total number of units of each resource type is given by the vector (5,8,15). The current allocation of resources is given by the matrix

$$A = \begin{bmatrix} 0 & 1 & 4 \\ 2 & 0 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 3 \end{bmatrix}$$

a. Determine if the current state of the system is safe.
b. Determine if a request by process 1 for 1 unit of resource 1 can be safely granted.
c. Determine if a request by process 2 for 4 units of resource 2 can be safely granted.

(**Note: Processes and Resources start with index 0**). You must show your work and justify each of your answers.

a) Total Allocation [0] = 0 + 2 + 1 + 1 = 4
Total Allocation [1] = 1 + 0 + 2 + 0 = 3
Total Allocation [2] = 4 + 1 + 1 + 3 = 9

Total Allocation [4, 3, 9]

Available = [5, 8, 15] - [4, 3, 9] = [1, 5, 6]

Need Matix = Max [i][j] - Current [i] [j]

Need			Order	Released			
4	0	0	3rd	4	5	10	[4, 5, 10] + [0, 1, 4]
1	1	3	1st	1	5	6	[1, 5, 6] + [2, 0, 1]
4	4	12	4th	4	6	14	
0	1	3	2nd	3	5	7	[3, 5, 7] + [1, 0, 3]

Since all processes are able to finish the system is safe.

b) Request[1] = [1, 0, 0]

[1, 0, 0] ≤ [1, 1, 3] - Need

[1, 0, 0] ≤ [1, 5, 6] - Available

New Available = [1, 5, 6] - [1, 0, 0] = [0, 5, 6]

New Allocation = [2, 0, 1] + [1, 0, 0] = [3, 0, 1]

New Need = [1, 1, 3] - [1, 0, 0] = [0, 1, 3]

Need	Order	Released	
4 0 0	3rd	4 5 10	[4, 5, 10] + [0, 1, 4]
0 1 3	1st	0 5 6	[0, 5, 6] + [3, 0, 1]
4 4 12	4th	4 6 14	
0 1 3	2nd	3 5 7	[3, 5, 7] + [1, 0, 3]

Since all processes can finish, the request by process 1 for 1 unit of resource 1 can be safely granted.

c) Request[2] = [0, 0, 4]

[0, 0, 4] ≤ [4, 4, 12] - Need

[0, 0, 4] ≤ [1, 5, 6] - Available

New Available = [1, 5, 6] - [0, 0, 4] = [1, 5, 2]

New Allocation = [1, 2, 1] + [0, 0, 4] = [1, 2, 5]

New Need = [4, 4, 12] - [0, 0, 4] = [4, 4, 8]

Need	Order	Released	
4 0 0	3rd	5 5 6	[5, 5, 6] + [0, 1, 4]
0 1 3	1st	1 5 2	[1, 5, 2] + [3, 0, 1]
4 4 8	4th	5 6 10	
0 1 3	2nd	4 5 3	[4, 5, 3] + [1, 0, 3]

Since all processes can finish, the request by process 2 for 4 units of resource 2 can be safely granted.