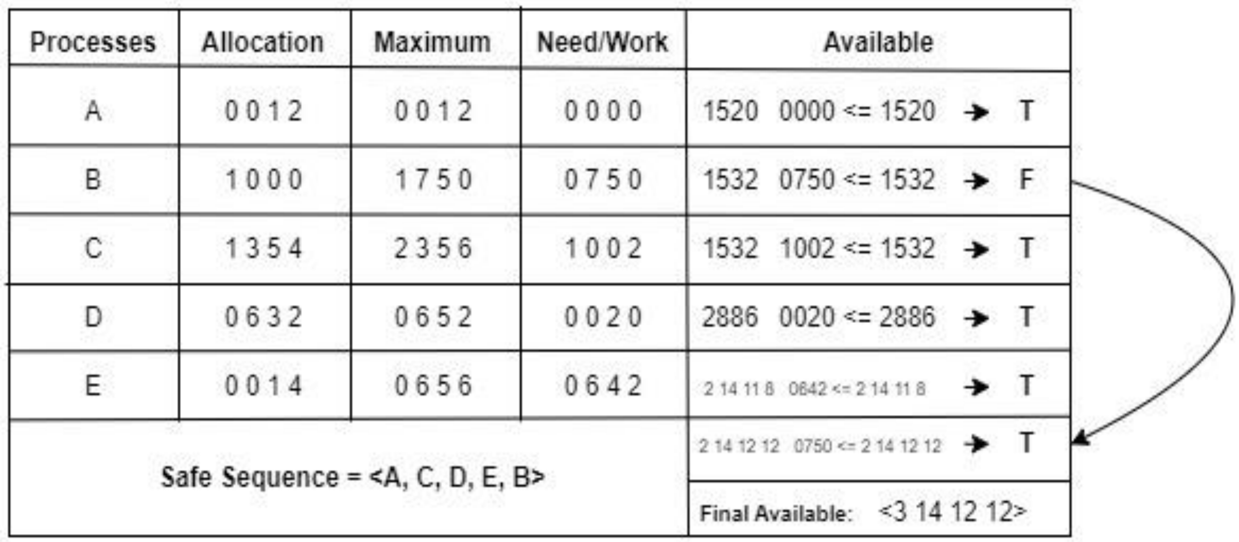
CNG 334

Assignment 2

Raed Alsheikh Amin 2528271

Task 1:

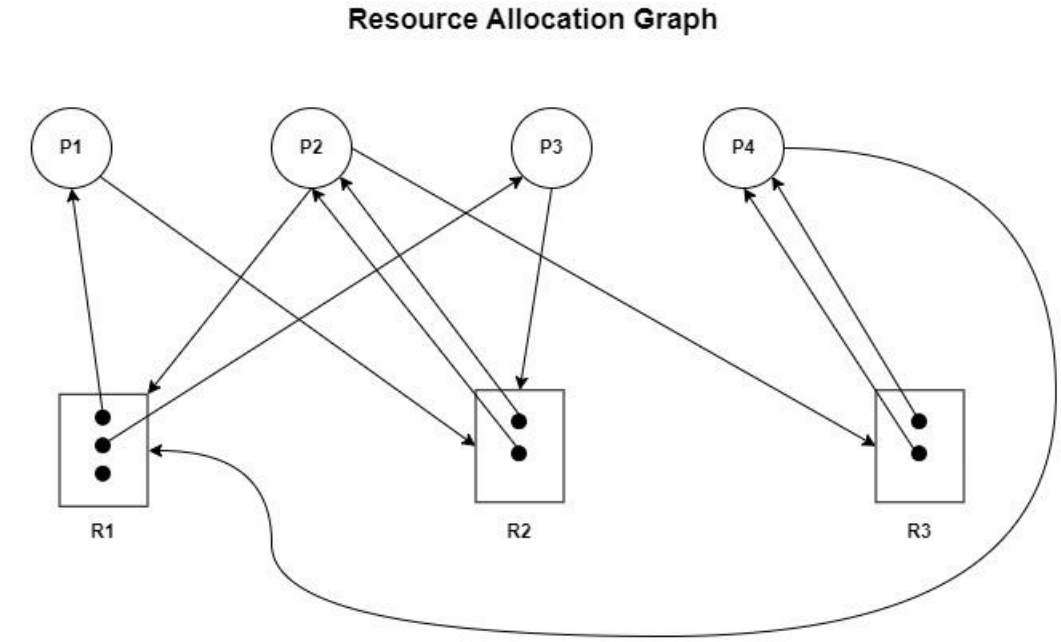


Processes	Allocation	Maximum	Need/Work	Available
A	0 0 1 2	0 0 1 2	0 0 0 0	1520 0000 <= 1520 → T
B	1 0 0 0	1 7 5 0	0 7 5 0	1532 0750 <= 1532 → F
C	1 3 5 4	2 3 5 6	1 0 0 2	1532 1002 <= 1532 → T
D	0 6 3 2	0 6 5 2	0 0 2 0	2886 0020 <= 2886 → T
E	0 0 1 4	0 6 5 6	0 6 4 2	2 14 11 8 0642 <= 2 14 11 8 → T
Safe Sequence = <A, C, D, E, B>				2 14 12 12 0750 <= 2 14 12 12 → T
				Final Available: <3 14 12 12>

The safe sequence is done in order => for example checking A valid , B invalid, C valid, D valid, E valid, then checking B valid again since we have a valid resources.

Task 2:

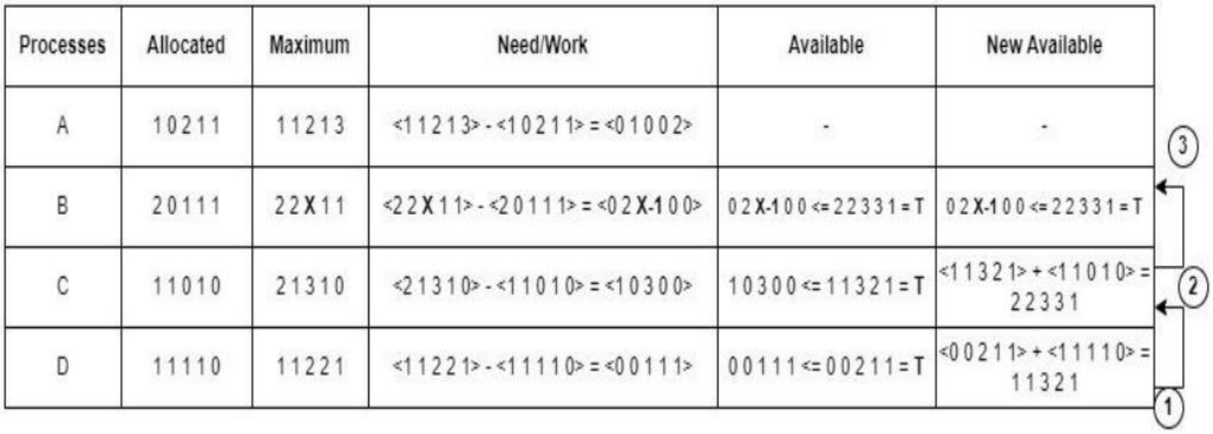
RAG:



There is **No Deadlock** illustrated in this system. We'll start with P4, as P4 executes first as one instance of R1 is available from the start. P4 releases both instances of R3 , and one instance of R1 in this way after being executed. P2 executes next as one instance of R1 and R3 are available as per its needs. After execution, P2 releases both instances of R2 after execution. P1 now has one instance of R1 available/free and one of R2 so it executes next. After execution, P1 releases one instance of R1. Finally, P3 needs one instance of R2 and P1 needs one instance of R2 to execute which are both available, so they both execute simultaneously without any deadlocks.

Task 3:

Available= 0 0 2 1 1



Processes	Allocated	Maximum	Need/Work	Available	New Available
A	10211	11213	<11213> - <10211> = <01002>	-	-
B	20111	22X11	<22X11> - <20111> = <02X-100>	02X-100 <= 22331 = T	02X-100 <= 22331 = T
C	11010	21310	<21310> - <11010> = <10300>	10300 <= 11321 = T	<11321> + <11010> = 22331
D	11110	11221	<11221> - <11110> = <00111>	00111 <= 00211 = T	<00211> + <11110> = 11321

First we compare the available resources with the need, and we check if we can satisfy or not

Since we are looking at DCBA safe sequence => we should go with that order.

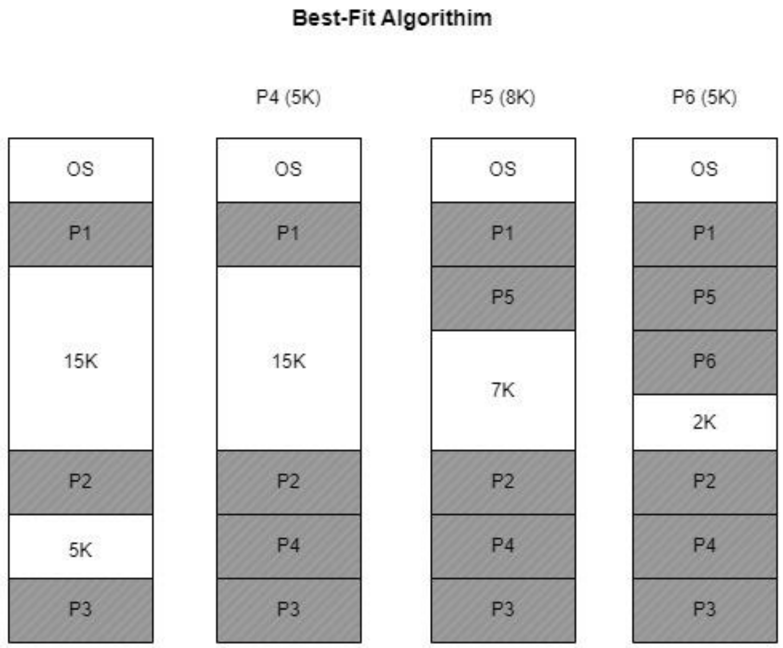
The table explains step 1 and 2
for step 3 => we know already that DCBA is a safe sequence, so X can't violate the available resources => X-1=3 => **X=4**

No need for finding A since the question is asking for the largest value of X in DCBA sequence

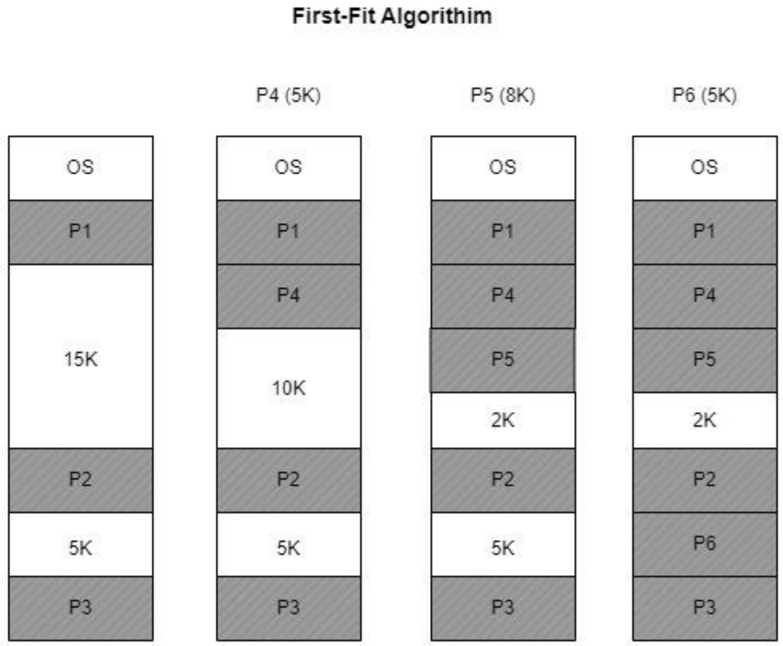
Task 4:

P4=5k, P5=8k and P6=5k

Best fit:

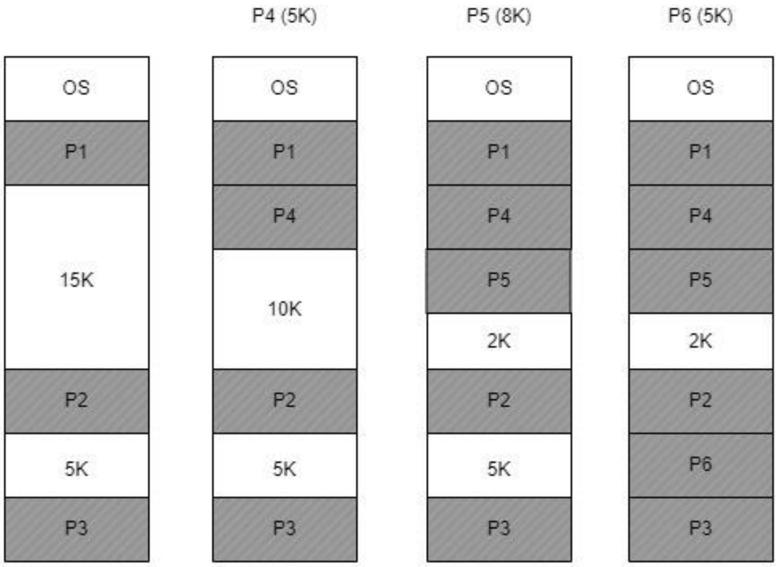


First fit:


First-Fit Algorithim

Worst fit:

Worst-Fit Algorithim



Task 5:

FIFO:

2	1	4	3	1	2	4	3	2	1

2	2	2	3	3	3	3	3	3	3
–	1	1	1	1	2	2	2	2	2
–	–	4	4	4	4	4	4	4	1

Queue:

~~2~~	~~1~~	~~4~~	3	2	1				

Page faults = 6

LRU:

2	1	4	3	1	2	4	3	2	1

2	2	2	1	4	3	1	2	4	3
–	1	1	4	3	1	2	4	3	2
–	–	4	3	1	2	4	3	2	1

Page fault= 8

Task 6:

1. **Optimal Replacement**
    - Ranking: Perfect
    - Belady's Anomaly: No
    - Explanation: Because it replaces pages that won't be needed for a long time, this algorithm has the lowest page-fault rate. It avoids Belady's anomaly and offers the best solution because it makes use of future knowledge.
2. **Least Recently Used Replacement**
    - Ranking: Good
    - Belady's Anomaly: No
    - Explanation: LRU provides a good approximation of the optimal replacement by replacing the page that has not been used for the longest. Although it experiences more page faults than OPT, overall performance is quite high, and Belady's anomaly is omitted.
3. **Second-Chance Replacement**
    - Ranking: Fair
    - Belady's Anomaly: No
    - Explanation: Second-chance is a FIFO variant that allows pages to be referenced and given another chance. While generally performing worse than LRU, it outperforms FIFO. Because it incorporates an extra mechanism to account for page references, it avoids Belady's anomaly.
4. **First-In, First-Out (FIFO) Replacement**
    - Ranking: Bad
    - Belady's Anomaly: Yes
    - Explanation: Regardless of how recently or frequently a page has been used, FIFO replaces the oldest one. Belady's anomaly, which states that an increase in frames can result in an increase in page faults, can cause higher page-fault rates in situations like this.