COMP 2432 Operating Systems

Solution to Written Assignment 2

1. Page Replacement.

(*a*) FIFO (3 frames): 19 page faults (page faults are indicated by *).

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	3	2*	2	2	2	2	2	2	5*	5	5	3*	3	3	3	3	4*	4	4	1*
-	1*	1	1	4*	4	4	4	3*	3	3	3	3	3	6*	6	6	2*	2	2	2	2	3*	3	3
-	-	2*	2	2	5*	5	5	5	5	5	4*	4	4	4	1*	1	1	1	0*	0	0	0	2*	2

(*b*) Optimal (3 frames): 14 page faults (3 possible answers on the last page fault).

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	5*	5	5	5	5	5	5	5	5	6*	1*	1	1	1	0*	0	4*	4	4	1*
-	1*	1	1	4*	4	4	4	3*	3	3	4*	4	4	4	4	3*	3	3	3	3	3	3	3	3
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	5*	5	5	5	5	5	5	5	5	6*	1*	1	1	1	0*	0	4*	4	4	4
-	1*	1	1	4*	4	4	4	3*	3	3	4*	4	4	4	4	3*	3	3	3	3	3	3	3	1*
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	5*	5	5	5	5	5	5	5	5	6*	1*	1	1	1	0*	0	4*	4	4	4
-	1*	1	1	4*	4	4	4	3*	3	3	4*	4	4	4	4	3*	3	3	3	3	3	3	3	3
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1*

(*c*) LRU (3 frames): 18 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	3	2*	2	2	2	2	2	2	2	2	1*	1	1	1	1	1	4*	4	4	1*
-	1*	1	1	4*	4	4	4	4	4	5*	5	5	5	5	5	3*	3	3	0*	0	0	3*	3	3
-	-	2*	2	2	5*	5	5	3*	3	3	4*	4	4	6*	6	6	2*	2	2	2	2	2	2	2
Stack content																								
-	-	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
-	1	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2
0	0	0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3

(*a*) FIFO (4 frames): 13 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	4*	4	4	4	4	4	4	4	4	4	4	4	3*	3	3	3	3	3	3	3	1*
-	1*	1	1	1	5*	5	5	5	5	5	5	5	5	5	5	5	2*	2	2	2	2	2	2	2
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	6*	6	6	6	6	0*	0	0	0	0	0
-	-	-	3*	3	3	3	3	3	3	3	3	3	3	3	1*	1	1	1	1	1	4*	4	4	4

(*b*) Optimal (4 frames): 10 page faults (4 possible answers on the last page fault).

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	4*	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	1*
-	1*	1	1	1	5*	5	5	5	5	5	5	5	5	6*	1*	1	1	1	0*	0	0	0	0	0
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-	-	-	3*	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

(*c*) LRU (4 frames): 14 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	4*	4	4	4	4	4	4	4	4	4	4	1*	1	1	1	1	1	1	3*	3	3
-	1*	1	1	1	5*	5	5	5	5	5	5	5	5	5	5	5	2*	2	2	2	2	2	2	2
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	3*	3	3	3	3	4*	4	4	4
-	-	-	3*	3	3	3	3	3	3	3	3	3	3	6*	6	6	6	6	0*	0	0	0	0	1*
Stack content																								
-	-	-	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
-	-	2	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2
-	1	1	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3
0	0	0	0	1	2	3	3	5	5	4	3	3	3	4	2	5	6	6	3	3	1	0	0	4

Note that for this reference string, LRU is better than FIFO with 3 frames, but FIFO is better than LRU with 4 frames.

(*d*) With respect to the number of page faults for FIFO (**19**) and LRU (**18**) in (*a*) and (*c*), it is possible to produce less than **19** page faults for FIFO, or less than **18** page faults for LRU by inserting an item into the reference string and there are more than one possible answers.
For example, the reference string could be modified into S_{11}, S_{12} or S_{13}, and the performance of LRU can be improved to **17** page faults. It can also be modified into S_{14}, S_{15} or S_{16}, and the performance of FIFO can be improved to **18** and **17** page faults.

S_{11}: 0 1 2 3 2 4 5 2 4 3 2 5 4 2 5 6 1 3 2 1 0 2 4 3 2 1 (LRU 17)

S_{12}: 0 1 2 3 4 2 5 2 4 3 2 5 4 2 5 6 1 3 2 1 0 2 4 3 2 1 (LRU 17)

S_{13}: 0 1 2 3 4 5 2 4 3 2 5 4 2 5 6 2 1 3 2 1 0 2 4 3 2 1 (LRU 17)

S_{14}: 0 1 2 3 4 5 2 4 3 2 5 4 2 5 6 1 2 3 2 1 0 2 4 3 2 1 (FIFO 18)

S_{15}: 0 1 4 2 3 4 5 2 4 3 2 5 4 2 5 6 1 3 2 1 0 2 4 3 2 1 (FIFO 17)

S_{16}: 0 1 2 5 3 4 5 2 4 3 2 5 4 2 5 6 1 3 2 1 0 2 4 3 2 1 (FIFO 17)

LRU on S_{11} (3 frames): 17 page faults.

0	1	2	3	2	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	3*	3	3	5*	5	5	3*	3	3	4*	4	4	6*	6	6	2*	2	2	2	2	2	2	2
-	1*	1	1	1	4*	4	4	4	4	4	5*	5	5	5	5	5	3*	3	3	0*	0	0	3*	3	3
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	1*	1	1	1	1	1	4*	4	4	1*

(*e*) With respect to the number of page faults for FIFO and LRU in (*a*) and (*c*), it is possible to produce non-decrasing total number of page faults of 37 by deleting an item from the reference string and there are more than one possible answers.
For example, the reference string could be modified into S_{21}, and the performance of LRU can be worsen to **19** page faults, with a total of 38 for FIFO and LRU. It can also be modified into S_{22}, S_{23}, S_{24} or S_{25}, and the total number of page faults remains at 37, without any increase.

S_{21}: 0 1 2 3 4 5 2 4 3 2 5 4 2 5 6 1 3 2 1 0 ~~2~~ 4 3 2 1 (LRU 19, total 38)

S_{22}: 0 1 2 3 4 5 2 4 3 ~~2~~ 5 4 2 5 6 1 3 2 1 0 2 4 3 2 1 (total 37)

S_{23}: 0 1 2 3 4 5 2 4 3 2 5 4 ~~2~~ 5 6 1 3 2 1 0 2 4 3 2 1 (total 37)

S_{24}: 0 1 2 3 4 5 2 4 3 2 5 4 2 5 6 ~~1~~ 3 2 1 0 2 4 3 2 1 (total 37)

S_{25}: 0 1 2 3 4 5 2 4 3 2 5 4 2 5 6 1 3 2 ~~1~~ 0 2 4 3 2 1 (total 37)

LRU on S_{21} (3 frames): 19 page faults

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	4	3	2	1
0*	0	0	3*	3	3	2*	2	2	2	2	2	2	2	2	1*	1	1	1	1	1	3*	3	3
-	1*	1	1	4*	4	4	4	4	4	5*	5	5	5	5	5	3*	3	3	0*	0	0	2*	2
-	-	2*	2	2	5*	5	5	3*	3	3	4*	4	4	6*	6	6	2*	2	2	4*	4	4	1*

2. Page Replacement.

Let us construct the list of predicted future with respect to each reference page access. Note that you *do not need* to produce the full list of predicted future, nor are the lists required in your answer. They are here merely to help determining the frame to be replaced as working steps. Normally just the first few entries will be sufficient for you in the page replacement algorithm. Note that wrap-around periods of access history are highlighted in alternating colors and all the "triangles" are repeating themselves with period of 10 and 5 respectively.

Reference	Predicted future ($P = 10$) to determine page replacement	Predicted future ($P = 5$) to determine page replacement
0	0 [actually not needed, since frames are not full]	0 [actually not needed, since frames are not full]
1	0 1 [actually not needed, since frames are not full]	0 1 [actually not needed, since frames are not full]
2	0 1 2 [actually not needed, since frames are not full]	0 1 2 [actually not needed, since frames are not full]
3	0 1 2 3	0 1 2 3
4	0 1 2 3 4	0 1 2 3 4
5	0 1 2 3 4 5	1 2 3 4 5 0
2	0 1 2 3 4 5 2	2 3 4 5 2 0 1
4	0 1 2 3 4 5 2 4	3 4 5 2 4 0 1 2
3	0 1 2 3 4 5 2 4 3	4 5 2 4 3 0 1 2 3
2	0 1 2 3 4 5 2 4 3 2	5 2 4 3 2 0 1 2 3 4
5	1 2 3 4 5 2 4 3 2 5 0	2 4 3 2 5 1 2 3 4 5 0
4	2 3 4 5 2 4 3 2 5 4 0 1	4 3 2 5 4 2 3 4 5 2 0 1
2	3 4 5 2 4 3 2 5 4 2 0 1 2	3 2 5 4 2 3 4 5 2 4 0 1 2
5	4 5 2 4 3 2 5 4 2 5 0 1 2 3	2 5 4 2 5 4 5 2 4 3 0 1 2 3
6	5 2 4 3 2 5 4 2 5 6 0 1 2 3 4	5 4 2 5 6 5 2 4 3 2 0 1 2 3 4
1	2 4 3 2 5 4 2 5 6 1 0 1 2 3 4 5	4 2 5 6 1 2 4 3 2 5 1 2 3 4 5 0
3	4 3 2 5 4 2 5 6 1 3 0 1 2 3 4 5 2	2 5 6 1 3 4 3 2 5 4 2 3 4 5 2 0 1
2	3 2 5 4 2 5 6 1 3 2 0 1 2 3 4 5 2 4	5 6 1 3 2 3 2 5 4 2 3 4 5 2 4 0 1 2
1	2 5 4 2 5 6 1 3 2 1 0 1 2 3 4 5 2 4 3	6 1 3 2 1 2 5 4 2 5 4 5 2 4 3 0 1 2 3
0	5 4 2 5 6 1 3 2 1 0 0 1 2 3 4 5 2 4 3 2	1 3 2 1 0 5 4 2 5 6 5 2 4 3 2 0 1 2 3 4
2	4 2 5 6 1 3 2 1 0 2 1 2 3 4 5 2 4 3 2 5 0	3 2 1 0 2 4 2 5 6 1 2 4 3 2 5 1 2 3 4 5 0
4	2 5 6 1 3 2 1 0 2 4 2 3 4 5 2 4 3 2 5 4 0 1	2 1 0 2 4 2 5 6 1 3 4 3 2 5 4 2 3 4 5 2 0 1
3	5 6 1 3 2 1 0 2 4 3 3 4 5 2 4 3 2 5 4 2 0 1 2	1 0 2 4 3 5 6 1 3 2 3 2 5 4 2 3 4 5 2 4 0 1 2
2	6 1 3 2 1 0 2 4 3 2 4 5 2 4 3 2 5 4 2 5 0 1 2 3	0 2 4 3 2 6 1 3 2 1 2 5 4 2 5 4 5 2 4 3 0 1 2 3
1	1 3 2 1 0 2 4 3 2 1 5 2 4 3 2 5 4 2 5 6 0 1 2 3 4	2 4 3 2 1 1 3 2 1 0 5 4 2 5 6 5 2 4 3 2 0 1 2 3 4

Based on the predicted future, one can execute the KBS algorithm using the predicted as the future in the Optimal algorithm. Just like LRU is an approximation to Optimal in attempting to predict the future, KBS is another approximation to Optimal in using another way to predict the future. The KBS results are shown below.

KBS, $P = 10$ (3 frames): 20 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	0	0	0	0	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	1*
-	1*	1	1	1	1	1	1	1	1	1	4*	4	4	6*	1*	3*	3	1*	0*	0	4*	3*	3	3
-	-	2*	3*	4*	5*	2*	4*	3*	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

KBS, $P = 5$ (3 frames): 16 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	0*	0	4*	3*	3	3
-	1*	1	1	1	1	2*	2	3*	2*	2	2	2	2	6*	1*	1	1	1	1	1	1	1	1	1
-	-	2*	3*	4*	4	4	4	4	4	4	4	4	4	4	4	3*	2*	2	2	2	2	2	2	2

Note that the *same* list of predicted future in the table above can be *reused* even when the number of available frames is changed (e.g. from $f = 3$ to $f = 4$). In fact, the predicted future is *independent* on the number of frames.

KBS, $P = 10$ (4 frames): 17 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	0	0	0	0	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	1*
-	1*	1	1	1	1	1	1	1	1	1	4*	4	4	4	4	4	4	4	4	4	4	3*	3	3
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-	-	-	3*	4*	5*	5	4*	3*	3	3	3	3	3	6*	1*	3*	3	1*	0*	0	0	0	0	0

KBS, $P = 5$ (4 frames): 13 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	0*	0	0	0	0	0
-	1*	1	1	1	1	1	1	3*	3	3	3	3	3	6*	1*	1	1	1	1	1	1	1	1	1
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-	-	-	3*	4*	4	4	4	4	4	4	4	4	4	4	4	3*	3	3	3	3	4*	3*	3	3

It can be seen that the performance of KBS depends on the period *P*. In our exercise, KBS (20 and 17 for *P*=10 and 16 and 13 for *P*=5) performs on average similar to LRU (18 and 14 with 3 and 4 frames) in terms of page fault count. Furthermore, LRU is easier to apply than KBS and there is no need to consider an extra parameter *P*, which could well affect the performance as indicated by this exercise (20 vs 16 and 17 vs 13 for 3 and 4 frames). It is obvious that *P*=5 yields much better performance. More scenarios would need to be studied for the general performance of the algorithms. In fact, *P*=5 is best and *P*=12 is worst for page fault performance.

An alternative formulation of KBS history on the position of the reflector.

Reference	Predicted future (*P* = 10) to determine page replacement	Predicted future (*P* = 5) to determine page replacement
0	- [actually not needed, since frames are not full]	- [actually not needed, since frames are not full]
1	0 [actually not needed, since frames are not full]	0 [actually not needed, since frames are not full]
2	0 1 [actually not needed, since frames are not full]	0 1 [actually not needed, since frames are not full]
3	0 1 2	0 1 2
4	0 1 2 3	0 1 2 3
5	0 1 2 3 4	0 1 2 3 4
2	0 1 2 3 4 5	1 2 3 4 5 0
4	0 1 2 3 4 5 2	2 3 4 5 2 0 1
3	0 1 2 3 4 5 2 4	3 4 5 2 4 0 1 2
2	0 1 2 3 4 5 2 4 3	4 5 2 4 3 0 1 2 3
5	0 1 2 3 4 5 2 4 3 2	5 2 4 3 2 0 1 2 3 4
4	1 2 3 4 5 2 4 3 2 5 0	2 4 3 2 5 1 2 3 4 5 0
2	2 3 4 5 2 4 3 2 5 4 0 1	4 3 2 5 4 2 3 4 5 2 0 1
5	3 4 5 2 4 3 2 5 4 2 0 1 2	3 2 5 4 2 3 4 5 2 4 0 1 2
6	4 5 2 4 3 2 5 4 2 5 0 1 2 3	2 5 4 2 5 4 5 2 4 3 0 1 2 3
1	5 2 4 3 2 5 4 2 5 6 0 1 2 3 4	5 4 2 5 6 5 2 4 3 2 0 1 2 3 4
3	2 4 3 2 5 4 2 5 6 1 0 1 2 3 4 5	4 2 5 6 1 2 4 3 2 5 1 2 3 4 5 0
2	4 3 2 5 4 2 5 6 1 3 0 1 2 3 4 5 2	2 5 6 1 3 4 3 2 5 4 2 3 4 5 2 0 1
1	3 2 5 4 2 5 6 1 3 2 0 1 2 3 4 5 2 4	5 6 1 3 2 3 2 5 4 2 3 4 5 2 4 0 1 2
0	2 5 4 2 5 6 1 3 2 1 0 1 2 3 4 5 2 4 3	6 1 3 2 1 2 5 4 2 5 4 5 2 4 3 0 1 2 3
2	5 4 2 5 6 1 3 2 1 0 0 1 2 3 4 5 2 4 3 2	1 3 2 1 0 5 4 2 5 6 5 2 4 3 2 0 1 2 3 4
4	4 2 5 6 1 3 2 1 0 2 1 2 3 4 5 2 4 3 2 5 0	3 2 1 0 2 4 2 5 6 1 2 4 3 2 5 1 2 3 4 5 0
3	2 5 6 1 3 2 1 0 2 4 2 3 4 5 2 4 3 2 5 4 0 1	2 1 0 2 4 2 5 6 1 3 4 3 2 5 4 2 3 4 5 2 0 1
2	5 6 1 3 2 1 0 2 4 3 3 4 5 2 4 3 2 5 4 2 0 1 2	1 0 2 4 3 5 6 1 3 2 3 2 5 4 2 3 4 5 2 4 0 1 2
1	6 1 3 2 1 0 2 4 3 2 4 5 2 4 3 2 5 4 2 5 0 1 2 3	0 2 4 3 2 6 1 3 2 1 2 5 4 2 5 4 5 2 4 3 0 1 2 3

KBS, *P* = 10 (3 frames): 21 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	0	0	0	0	0	4*	4	4	4	4	4	4	1*	1	1	1	1	1	1
-	1*	1	1	1	1	1	1	1	1	1	1	2*	2	6*	1*	3*	3	3	0*	0	4*	3*	3	3
-	-	2*	3*	4*	5*	2*	4*	3*	2*	5*	5	5	5	5	5	5	2*	2	2	2	2	2	2	2

KBS, *P* = 5 (3 frames): 18 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	2*	2	3*	2*	2	2	2	2	2	2	2	2	1*	1	1	4*	4	4	4
-	1*	1	1	1	1	1	4*	4	4	4	4	4	4	6*	1*	3*	3	3	3	3	3	3	3	1*
-	-	2*	3*	4*	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	0*	2*	2	2	2	2

KBS, *P* = 10 (4 frames): 17 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	0	0	0	0	0	4*	4	4	4	4	4	4	1*	1	1	1	1	1	1
-	1*	1	1	1	1	1	1	1	1	1	1	1	1	6*	1*	3*	3	3	0*	0	4*	3*	3	3
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-	-	-	3*	4*	5*	5	4*	3*	3	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	5

KBS, *P* = 5 (4 frames): 14 page faults.

0	1	2	3	4	5	2	4	3	2	5	4	2	5	6	1	3	2	1	0	2	4	3	2	1
0*	0	0	0	0	0	0	0	3*	3	3	3	3	3	6*	1*	3*	3	3	3	3	3	3	3	3
-	1*	1	1	1	1	1	4*	4	4	4	4	4	4	4	4	4	4	1*	1	1	1	1	1	1
-	-	2*	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-	-	-	3*	4*	5*	5	5	5	5	5	5	5	5	5	5	5	5	5	0*	0	4*	4	4	4

It can be seen that the performance of KBS depends on the period *P*. In our exercise, KBS (21 and 17 for *P*=10 and 18 and 14 for *P*=5) performs worse or similar to LRU (18 and 14 with 3 and 4 frames) in terms of page fault count. Furthermore, LRU is easier to apply than KBS and there is no need to consider an extra parameter *P*, which could well affect the performance as indicated by this exercise. *P*=10 leads to a rather poor performance. More scenarios would need to be studied for the general performance of the algorithms. In fact, *P*=4 will yield the best performance and *P*=13 the worst.

3. Deadlock Avoidance.

(*a*) Total number of resources = (**10 6 6 8**) and *Avail* = (1 1 1 0). There are sufficient resources for the request. Pretend that the request by P_2 is granted, *Avail* = (1 0 1 0). The revised resource allocation state would be:

	Allocation				Max				Need			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	1	0	1	2	2	2	2	2	1	2	1	0
P_1	2	1	0	1	3	1	1	1	1	0	1	0
P_2	1	3	0	1	2	4	3	2	1	1	3	1
P_3	2	1	1	0	3	2	1	2	1	1	0	2
P_4	1	1	1	2	2	3	2	3	1	2	1	1
P_5	2	0	2	2	3	1	2	2	1	1	0	0

The resultant state is a *safe state*. A possible safe sequence is $<P_1, P_5, P_2, P_0, P_3, P_4>$.
Start from *Work* = (1 0 1 0).
P_1 can finish, *Work* = (1 0 1 0) + (2 1 0 1) = (3 1 1 1).
P_5 can finish, *Work* = (3 1 1 1) + (2 0 2 2) = (5 1 3 3).
P_2 can finish, *Work* = (6 4 3 4).
P_0 can finish, *Work* = (7 4 4 6).
P_3 can finish, *Work* = (9 5 5 6).
P_4 can finish, *Work* = (**10 6 6 8**). So all processes could finish. Thus, the request can be ***granted***.
There are a total of **12** possible safe sequences.

The full list is $<P_1, P_5, P_2, P_0, P_3, P_4>$, $<P_1, P_5, P_2, P_0, P_4, P_3>$, $<P_1, P_5, P_2, P_3, P_0, P_4>$, $<P_1, P_5, P_2, P_3, P_4, P_0>$,$<P_1, P_5, P_2, P_4, P_0, P_3>$, $<P_1, P_5, P_2, P_4, P_3, P_0>$, $<P_1, P_5, P_3, P_0, P_2, P_4>$, $<P_1, P_5, P_3, P_0, P_4, P_2>$, $<P_1, P_5, P_3, P_2, P_0, P_4>$, $<P_1, P_5, P_3, P_2, P_4, P_0>$, $<P_1, P_5, P_3, P_4, P_0, P_2>$, and $<P_1, P_5, P_3, P_4, P_2, P_0>$. This full list is not necessary in your answer.

(*b*) Here we are to find a resource type *X* such that by increasing Max_x and $Need_x$ by **two** for each possible process P_x, all the adjusted resource allocation states remain safe. We also know that $x \neq 1$.

Let *X* be *B*. Consider P_5. Max_5 = (3 3 2 2). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
Let *X* be *C*. Consider P_5. Max_5 = (3 1 4 2). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
Let *X* be *D*. Consider P_5. Max_5 = (3 1 2 4). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
In fact, if *X* is *A*, it can be verified that there is *a safe sequence for each of the cases* that any process, except for P_1, under-reports its need for *A* by two.

Thus, ***X* is *A***.

(*c*) Here we are to find a process P_y ($y \neq 1$) such that by by increasing Max_y and $Need_y$ by one for two resource Y_1 and Y_2, the adjusted resource allocation state becomes unsafe for any Y_1 and Y_2.

Let P_y be P_5 and Y_1 be *B*. Consider $Y_2 = A$. Max_5 = (4 2 2 2). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
Let P_y be P_5 and Y_1 be *B*. Consider $Y_2 = C$. Max_5 = (3 2 3 2). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
Let P_y be P_5 and Y_1 be *B*. Consider $Y_2 = D$. Max_5 = (3 2 2 3). It can be found that the resultant state is *unsafe*.
Start from *Work* = (1 0 1 0). P_1 can finish, *Work* = (3 1 1 1). Now no process can finish and the state is *unsafe*.
As a result, if $y = 5$ and $Y_1 = B$, the system state is *unsafe* regardless of Y_2.

Note that even if $y = 5$, $Y_1 = A$ and $Y_2 = C$ will lead to a *safe* state.
Max_5 = (4 1 3 2). Start from *Work* = (1 0 1 0).
P_1 can finish, *Work* = (3 1 1 1). P_5 can finish, *Work* = (5 1 3 3). P_2 can finish, *Work* = (6 4 3 4).
P_0 can finish, *Work* = (7 4 4 6). P_3 can finish, *Work* = (9 5 5 6). P_4 can finish, *Work* = (**10 6 6 8**).
So all processes could finish and the state is *safe*.
Similarly, $y = 5$, $Y_1 = A$ and $Y_2 = D$ will lead to a *safe* state.
Also, $y = 5$, $Y_1 = C$ and $Y_2 = D$ will lead to a *safe* state.
Finally, it can be verified that if $y = 0$, 2, 3 or 4, the system state remains safe regardless of Y_1 and Y_2.

Thus, ***P_y* is P_5, and *Y_1* is *B***.

4. Deadlock Detection.

(*a*) Total number of resources = (**10 4 6 5**). The state is not a deadlocked state. There is a completion sequence. A possible completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>.
Start from *Work* = (1 0 1 0).
P_2 can finish, *Work* = (2 1 1 0).
P_4 can finish, *Work* = (3 2 2 1).
P_1 can finish, *Work* = (5 2 2 2).
P_0 can finish, *Work* = (6 2 3 2).
P_3 can finish, *Work* = (8 2 4 4).
P_5 can finish, *Work* = (**10 4 6 5**).
Now, all processes could finish successfully and there is *no deadlock*.
There are a total of **6** possible completion sequences.

The full list is <P_2, P_4, P_1, P_0, P_3, P_5>, <P_2, P_4, P_1, P_0, P_5, P_3>, <P_2, P_4, P_1, P_3, P_0, P_5>, <P_2, P_4, P_1, P_3, P_5, P_0>, <P_2, P_4, P_1, P_5, P_0, P_3>, <P_2, P_4, P_1, P_5, P_3, P_0>. This full list is not necessary in your answer.

(*b*) Here we are to find P_x such that there is a deadlocked state if the request by P_x for any resource is changed.
Let x be 2, and check out for P_2.
Let X be A. Req_2 = (3 0 1 0). It is obvious that no process can complete and there is a deadlock.
Let X be B. Req_2 = (1 2 1 0). Again, no process can complete and there is a deadlock.
Let X be C. Req_2 = (1 0 3 0). No process can complete and there is a deadlock.
Let X be D. Req_2 = (1 0 1 2). No process can complete and there is a deadlock.
We can conclude that **P_x is P_2**.

We can verify that for other processes, the system is not in a deadlocked state for certain resource X.
Let P_x be P_0 and Y be A. Req_0 = (2 1 1 2). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>. There is no deadlock.
Let P_x be P_1 and Y be A. Req_1 = (3 1 1 1). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>. There is no deadlock.
Let P_x be P_3 and Y be A. Req_3 = (3 2 1 2). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>. There is no deadlock.
Let P_x be P_4 and Y be A. Req_4 = (2 1 1 0). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>. There is no deadlock.
Let P_x be P_5 and Y be A. Req_5 = (3 0 1 2). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>. There is no deadlock.
Therefore, x cannot be 0, 1, 3, 4, 5.

(*c*) Here we are to find P_y such that there is a deadlocked state if the requests by P_y on resource D and any other resource $Y_1 \neq D$ are changed. Furthemore, if both Y_1 and Y_2 are not D, the system is not deadlocked.

Let P_y be P_1, Y_1 be A, Y_2 be D. Req_1 = (2 1 1 2). Only P_2 and P_4 can finish, and other processes are deadlocked.
Let P_y be P_1, Y_1 be B, Y_2 be D. Req_1 = (1 2 1 2). Only P_2 and P_4 can finish, and other processes are deadlocked.
Let P_y be P_1, Y_1 be C, Y_2 be D. Req_1 = (1 1 2 2). Only P_2 and P_4 can finish, and other processes are deadlocked.
Let P_y be P_1, Y_1 be A, Y_2 be B. Req_1 = (2 2 1 1). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>: no deadlock.
Let P_y be P_1, Y_1 be A, Y_2 be C. Req_1 = (2 1 2 1). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>: no deadlock.
Let P_y be P_1, Y_1 be B, Y_2 be C. Req_1 = (1 2 2 1). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>: no deadlock.
We can conclude that **P_y is P_1**.

It can be easily verified that for y = 0, 3, 5, the system state will not be a deadlocked state for any pair of resource.
Let P_y be P_0, Y_1 be A, Y_2 be D. Req_0 = (1 1 1 3). A completion sequence is <P_2, P_4, P_1, P_3, P_0, P_5>: no deadlock.
Let P_y be P_3, Y_1 be A, Y_2 be D. Req_3 = (2 2 1 3). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>: no deadlock.
Let P_y be P_5, Y_1 be A, Y_2 be D. Req_5 = (2 0 1 3). A completion sequence is <P_2, P_4, P_1, P_0, P_3, P_5>: no deadlock.
It can also be verified for Y_1 and Y_2 being other resource.

It can be verified that for y = 2, 4, the system will be in a deadlocked state for any pair of resource.
Let P_y be P_2, Y_1 be A, Y_2 be D. Req_2 = (2 0 1 1). It is obvious that no process can complete and there is a deadlock.
Let P_y be P_4, Y_1 be A, Y_2 be D. Req_4 = (1 1 1 1). Again, no process can complete and there is a deadlock.
It can also be verified for Y_1 and Y_2 being other resource.

Therefore, y cannot be 0, 2, 3, 4, 5.