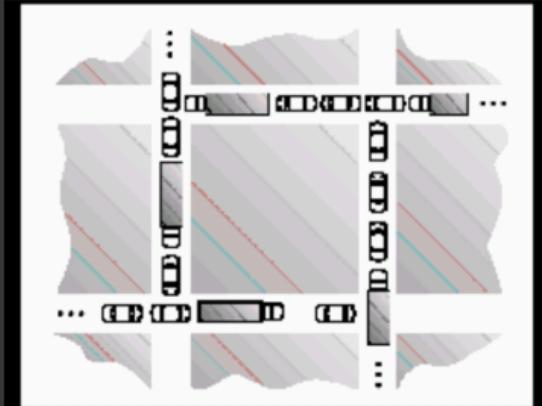




Exercises:

1. Consider the traffic deadlock depicted in the figure:

- Show that the four necessary conditions for deadlock indeed hold in this example.
- State a simple rule that will avoid deadlock in this system.



- 1) Mutual Exclusion
 - Each car needs access to the road in front of it. If two cars cannot occupy the same road at the same time, the road cant be shared hence mutual exclusion.
- 2) Hold and Wait
 - A lane is waiting for the cars in the other lane it merges to, to move.
- 3) No Preemption
 - A car cannot give up its position on the road. Since it cant be moved, a deadlock cannot be resolved.
- 4) Circular Wait
 - Similar to hold and wait, but a cycle. The north lane is waiting for the cars to move from the west lane, which is also waiting for cars to move from the south lane, which is also waiting for cars to move from the east lane, which is waiting for the north lane to move. This creates a circular wait.

Exercises:



2. Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free?

Since there are 4 identical resources and 3 processes that each need at most 2 of those resources. The system is deadlock free as there will always be an extra resource that a process can utilize, finish its task, and free its resource.

Exercises:



3. Cinderella and the Prince are getting divorced. To divide their property, they have agreed on the following algorithm. Every morning, each one may send a letter to the other's lawyer requesting one item of property. Since it takes a day for letters to be delivered, they agreed that if both discover that they have requested the same item on the same day, the next day they will send a letter canceling the request. Among their property is their dog (Woofer), doghouse, their canary (Tweety) and Tweety's cage.



The animals love their houses, so it has been agreed that any division of property separating an animal from its house is invalid, requiring the whole division to start over from scratch. Both Cinderella and the prince desperately want Woofer. So they can go on (separate) vacations, each spouse has programmed a personal computer to handle the negotiation. When they came back from vacation, the computers were still negotiating. Why? Is deadlock possible? Is starvation possible? Discuss.

No, it's not a deadlock, it's a livelock since not all parties are blocked but are just going on indefinitely.

- Since both parties want and prioritize to get Woofer, a livelock will happen.
- Both parties want Woofer, so each morning their program sends a request for Woofer.
- If both parties request at the same time on the same day for Woofer, they will send a cancellation on the next day.
- The next day, they will still request for Woofer as they again prioritize getting woofer, resulting in the same loop.
- Another thing that could cause a loop is when 1 party requests for Woofer and the other requests for the dog house, causing the division to restart.

There is also no starvation as starvation means that 1 side keeps on losing and the other side keeps on winning. In this case no one is winning, both are losing.



4. Consider the following and answer the questions that follow:

$$P = P_1, P_2, P_3, P_4, P_5$$

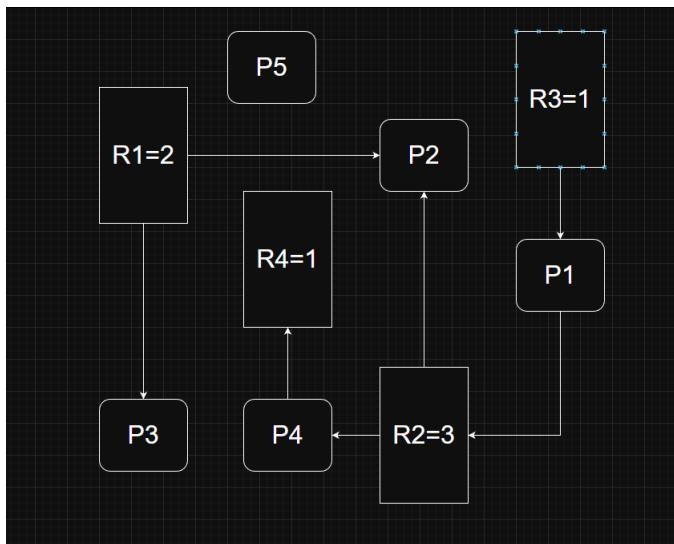
$$R = R_1, R_2, R_3, R_4$$

Instances: $R_1=2, R_2=3, R_3=1, R_4=1$

$$E=\{ R_3 \rightarrow P_1, P_1 \rightarrow R_2, R_2 \rightarrow P_2, R_2 \rightarrow P_4, R_1 \rightarrow P_2, R_1 \rightarrow P_3, P_4 \rightarrow R_4 \}$$

Answer the ff. questions:

- 1: Is there a cycle? (Y/N)
- 2: Is there a deadlock? (Y/N)
- 3: Will P_1 's request for R_2 be granted? (Y/N)
- 4: Will P_4 's request for R_4 be granted? (Y/N)
- 5: How many available instances are there for R_2 ?



1. Is there a cycle?
 - No
2. Is there a deadlock?
 - No
3. Will P_1 's request for R_2 be granted?
 - Yes it will be granted. There is still an available instance for P_1
4. Will P_4 's request for R_4 be granted?
 - Yes it will be granted, though R_4 only has 1 instance, it is not being used by any other process so P_4 's request will be granted.
5. How many available instances are there for R_2 ?
 - Considering P_1 's request for R_2 will be granted, there will be no available instances left to be allocated.



5. Eight (8) processes P_0 through P_7 with 4 resource types: A (10 instances), B (5), C (11) and D (6).

	<u>Allocation</u>				<u>Max</u>			
	A	B	C	D	A	B	C	D
P_0	0	1	2	1	7	4	9	2
P_1	2	1	0	0	3	2	2	0
P_2	0	0	2	0	9	0	7	2
P_3	2	1	0	2	2	2	4	3
P_4	0	0	2	0	4	3	6	3
P_5	2	0	2	0	2	1	3	2
P_6	2	1	1	1	5	2	2	1
P_7	1	0	2	0	7	4	8	5

Question: Is the system safe? Justify your answer.