   
   (a) Draw an approvability graph system in which action $a_1$ must be the first action and either action $a_2$ or $a_3$ is the second and final action. Assume that there is a separation of duty constraint between $a_1$ and $a_2$ and between $a_1$ and $a_3$.
   
   (b) Assuming each action is performed by the same role, how many users are required to be in the role so that approvability never gets stuck?
   
   (c) Does the number of users needed for approvability change if there is a different user constraint between $a_2$ and $a_3$? Explain why or why not.


   (a) Given the below approvability graph, draw in constraints which ensure that each task is performed by at least 2 users.

   (b) Do your constraints ensure that no more two users are ever needed to ensure that it does not get stuck? Explain.

   (c) What is the maximum number of users which can be involved in a task.

3. Security Properties: Confidentiality. Let $mayFlow(l, l')$ be the group of users who have permission to write $l'$ after having read $l$.

   (a) Consider the labels, $H, M_1, M_2, L$ in a diamond lattice and write out the associated mayFlows and the relations between them (that is, inclusion relations which must hold on the groups).

   (b) Now add a downgrade from $H$ to $M_1$ and also from $M_1$ to $L$ to your previous solution by defining additional mayFlows.

   (c) Is it possible using the previous solution to downgrade $H$ to $M_2$? If so, describe how; if not, explain why not.
4. Executable.

(a) Give examples of why you would want a particular to have (i) more permissions and (2) less permission than a user normally does.

(b) Under what conditions might the executable be the sole basis for permissions?

(c) Does Unix have a mechanism which allows permissions to be associated with executables? Explain.

5. Basis for permissions. We have seen three different issues which effect the permissions a process has:

(a) The user on whose behalf the process runs,

(b) The program which the process executes, and

(c) The past actions of the process.

The user and the developer both pose security threats to the system, but in what way are the past actions of the process relevant?

6. Reference monitor. The reference monitor does two things (1) for each mediated action it decides whether the action is allowed or denied and (2) it provides new labels.

The purpose of (1) is obvious, but give two examples of why (2) is needed?

7. HRU. What is customizable in the HRU access matrix?

8. Type Enforcement. Give the TE access matrix which implements the diamond lattice.

9. RBAC. Consider an access matrix whose rows represent users and columns represent objects.

(a) Show how the UA and PA can be composed to provide entries in the access matrix.

(b) Show how a singe change to the UA can result in multiple changes to the access matrix.

(c) Show how a singe change to the PA can result in multiple changes to the access matrix.