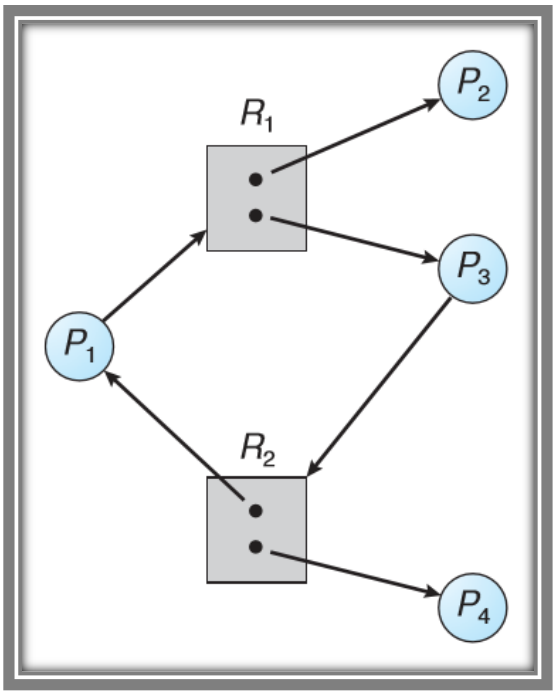
Cairo University
Faculty of Computers & Artificial Intelligence
Department of Computer Science

Artificial Intelligence Course
Assignment 1 – Part 1

Resource-Allocation Graph: *[4 marks]*

❖ *About the graph:*

The graph consists of a set of vertices and a set of edges. The vertices represent all the active processes in the system {P_1, P_2, ..., P_n} as well as all the resource types in the system {R_1, R_2, ..., R_m}. The dots found in each resource square represent the number of instances of that resource. A directed edge from process P_i to resource type R_j ($P_i \rightarrow R_j$) signifies that Pi has requested an instance of R_j and is currently waiting for that resource. A directed edge from resource type R_j to process P_i ($R_j \rightarrow P_i$) signifies that an instance of R_j has been allocated to P_i.



A process releases its allocated resources only when it acquires all the resources it needs (all its requests are fulfilled). So, if the graph contains no cycles, no process in the system is deadlocked. Otherwise, a deadlock may exist. If each resource type has exactly one instance, then a cycle implies that a deadlock has occurred. On the other hand, if each resource type has several instances, a cycle doesn't necessarily imply that a deadlock has occurred. In the previous figure, a cycle exists; however, there is no deadlock. This is because process P_4 can release its instance of resource R_2. That resource can then be allocated to P_3, breaking the cycle.

❖ *What you are required to do:*

You are required to make a prolog program that determines whether a graph can reach a safe state (with no deadlock) or not and if the graph can reach a safe state, you should output the process execution order that made the graph be in such a state. You will have to define the graph data as facts and define the necessary predicates/rules that detect deadlocks. **For simplicity, assume we only have 2 resource types (R_1 and R_2).**

❖ *A sample test case:*

This test case shows the sample input that should be entered and the sample output for the graph in the previous figure.

```
?- safe_state(X).
X = [p2, p4, p1, p3].
```

Notice that that graph has different solutions.

❖ *The input file:*

The input file should look somewhat like the opposite figure, but you can adjust the facts/predicates if you need to.

Problem(test).pl

```
1  process(p1).
2  process(p2).
3  process(p3).
4  process(p4).
5
6  resource(r1).
7  resource(r2).
8
9  allocated(p1, r2).
10 allocated(p2, r1).
11 allocated(p3, r1).
12 allocated(p4, r2).
13
14 requested(p1, r1).
15 requested(p3, r2).
16
17 available_instances([[r1, 0],[r2, 0]]).
```

Important Notes: *(Please read these notes carefully to avoid losing grades)*

- In this assignment, you will work in teams. **The minimum number of students in a team is 2 and the maximum is 3.**
- Please submit **one compressed folder** containing your **(.pl) file**. The folder name should follow this structure: **ID1_ID2_ID3_GX,Y**
- **Cheating students will take -6** and no excuses will be accepted. If you have any problems during the submission, contact your TA but don't, under any circumstances, give your code to your friends.

Grading Criteria:

Problem 2	
Suitable representation of graph info	**0.5**
Suitable predicates & rules	**2**
Output (true + execution order / false)	**1.5**
Bonus: Solve the problem for graphs with any number of resources.	**1**

Good luck