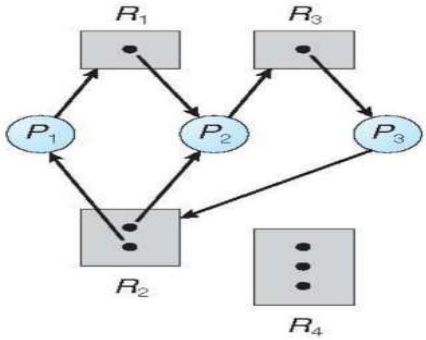
LAB # 10 Deadlock Detection and Avoidance using Banker's Algorithm

Task1:

Given the resource allocation graph below. Write the code to initialize the maximum instances of each resource using for loop.



//Data Structures

#define MAX_PROCESS 3

#define MAX_RESOURCES 3

Int max_instances[MAX_RESOURCES]

Int request[MAX_PROCESS][MAX_RESOURCES]

Int allocation[MAX_PROCESS][MAX_RESOURCES]

Task 2: write the code to initialize the request matrix according to the RAG shown and print the matrix.

Task 3:

Write the code to print the available resource(instances) vector.

Task 4:

Write the code to prompt the user to enter the maximum required instances of each resource by the each process. Also print the maximum required matrix.(Assume 3 processes and 3 resources). Make sure the maximum required instances by each process must not exceed maximum instances of each resource.

//Data Structure required/involved

Int Max_required[3][3];

Int max_instances[3];

Task 5:

Write the code to calculate the need matrix for the processes.(Assume 3 processes and 3 resources)

```
//Data Structure

Int need[3][3];

Int Max_required[3][3];

Int allocated[3][3];
```

Task 6:

Write the generic code for variable number of processes, resources and instances of each resource. Initialize the allocation matrix, maximum instance requirement matrix.

1. Enter the number of processes
2. Enter the number of resources
3. Initialize the resources with their maximum instances.
4. Allocate the resource instances to each process
5. Initialize the Maximum requirement of resource instances for each process
6. Print the maximum instances each resource has.
7. Print the allocation matrix
8. Print the maximum requirement matrix

Task 7:

Write the code to calculate the need matrix and Available resource instances vector.

Task 8:

Using Need matrix and available resource instance vector in the task 7. Identify and print the processes which will create deadlock if they started immediately.

Task 9:

using the initialized data structures in task 6 and task 7. Implement the Banker's Algorithm to print the safe sequence of processes to avoid the deadlock.