# COMP3500: The Banker’s Algorithm Part 1

**Exercise 1:** What do the following variables indicate? Can the request be honored? Why?

*Available[5] = 6;*

*Max* [*3, 5*] = *7*;

Allocation[*3, 5*] = *2;*

Need[3, 5] = \_\_\_\_\_\_?

**Exercise 2:** How many processes and resource types are there in the following example? Why Available[A]=3?

****

**Exercise 3:** Can you explain the following case, where *P*1 requests (1,0,2)?

(3.1) Can the request of P1 be granted? Why?

(3.2) How to update the “Allocation”, “Need”, and Available data?

 

**Exercise 4:** Can request for (3,3,0) by P4 be granted? Why? Can request for (0,2,0) by P0 be granted?

*Allocation Need Available*

*A B C A B C A B C*

*P*0 0 1 0 7 4 3 2 3 0

*P*1 3 0 2 0 2 0

*P*2 3 0 2 6 0 0

*P*3 2 1 1 0 1 1

*P*4 0 0 2 4 3 1

**Exercise 5:** The Resource-Request Algorithm for Process *Pi*? Requesti = request vector for process Pi. If Requesti [j] = k then process Pi wants k instances of resource type Rj

1. If Requesti \_\_\_\_\_\_\_\_\_\_ Needi go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim
2. If Requesti \_\_\_\_\_\_\_\_\_\_ Available, go to step 3. Otherwise Pi must wait, since resources are not available
3. Pretend to allocate requested resources to Pi by modifying the state as follows:

* Available = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
* Allocationi = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
* Needi = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

If safe: the resources are allocated to Pi

If unsafe: Pi must wait, and the old resource-allocation state is restored.