❖ Resource-Request Algorithm for Process P_i

This algorithm is used to determine whether requests can be safely granted.

Let $Request_i$ = request vector for process P_i

If $Request_i [j] == k$ then process P_i wants k instances of resource type R_j

P_i requests resources:

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

$$Available = Available - Request_i$$
$$Allocation_i = Allocation_i + Request_i$$
$$Need_i = Need_i - Request_i$$

- If safe ⇒the resources are allocated to P_i
- If unsafe ⇒P_i must wait, and the old resource-allocation state is restored

❖ Example of Banker's Algorithm

Consider a system with 5 processes P_0 through P_4; and 3 resource types:
A (10 instances), B (5 instances), and C (7 instances). Suppose that snapshot at time T_0:

	Allocation	*Max*	*Available*
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

The content of the matrix ***Need*** is defined to be ***Max*** - ***Allocation*** and is as follows:

	Need
	A B C
P_0	7 4 3
P_1	1 2 2
P_2	6 0 0
P_3	0 1 1
P_4	4 3 1

Available
3 / 3 / 2
P1 ok; releases 2 / 0 / 0
5 / 3 / 2
P3 ok; releases 2 / 1 / 1
7 / 4 / 3
P4 ok; releases 0 / 0 / 2
7 / 4 / 5
P2 ok ; releases 3 / 0 / 2
10 / 4 / 7
P0 ok ; releases 0 / 1 / 0
10 / 5 / 7

The system is in a safe state since the sequence **<P_1 , P_3 , P_4 , P_2 , P_0 >** satisfies safety criteria.

❖ **Example: P_1 Request (1,0,2)**

Suppose now that process P_1 requests 1 additional instance of resource type ***A*** and 2 instances of resource type ***C***, so $Request_1 = (1,0,2)$.

Check $\boldsymbol{Request_1} \leq \boldsymbol{Available}$: $(1,0,2) \leq (3,3,2)$, which is true. We then pretend that this request has been satisfied, and we arrive at the following new state:

	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	

To determine whether this new system state is safe we execute the safety algorithm and find that the sequence **<P_1, P_3, P_4, P_0, P_2>** satisfies the safety requirement. Hence, we can immediately grant the request of process P_1.

Exercise:

- Can request for (3,3,0) by P_4 be granted?
- Can request for (0,2,0) by P_0 be granted?