Banker's Algorithm

- Assume we have the following resources:
  5 tape drives
  2 graphic displays
  4 printers
  3 disks
  We can create a vector representing our total resources: Total = (5, 2, 4, 3).

- Consider we have already allocated these resources among four processes as demonstrated by the following matrix named **Allocation**.

Process	Tape drive	Graphic Dis	Printers	Disk
A	2	0	1	1
B	0	1	0	0
C	1	0	1	1
D	1	1	0	1

- The vector representing the allocated resources is the sum of these columns: **Allocated = (4, 2, 2, 3).**

- We also need a matrix to show the number of each resource still needed for each process; we call this matrix **Need**.

Process	Tape drive	Graphic Dis	Printers	Disk
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0

- The vector representing the available resources will be the sum of these columns subtracted from the Allocated vector: **Available = (1, 0, 2, 0)**.

<u>The algorithm:</u>

1. Find a row in the **Need** matrix which is less than the **Available** vector. If such a row exists, then the process represented by that row may complete with those additional resources. If no such row exists, eventual deadlock is possible.

2. You want to double check that granting these resources to the process for the chosen row will result in a safe state. Looking ahead, pretend that that process has acquired all its needed resources, executed, terminated, and returned resources to the **Available** vector. Now the value of the **Available** vector should be greater than or equal to the value it was previously.

3. Repeat steps 1 and 2 until all the processes have successfully reached pretended termination (this implies that the initial state was safe); or deadlock is reached (this implies the initial state was unsafe).

Please follow the following iterations:

Iteration 1:

- Examine the **Need** matrix. The only row that is less than the **Available** vector is the one for Process D.
  Need(Process D) = (0, 0, 1, 0) < (1, 0, 2, 0) = **Available**

- If we assume that Process D completes, it will turn over its currently allocated resources, incrementing the **Available** vector.

  (1, 0, 2, 0) Current value of Available
  \+ (1, 1, 0, 1) **Allocation** (Process D)
  ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `

  (2, 1, 2, 1) Updated value of **Available**

Iteration 2:

- Examine the **Need** matrix, ignoring the row for Process D. The only row that is less than the **Available** vector is the one for Process A.
  Need(Process A) = (1, 1, 0, 0) < (2, 1, 2, 1) = **Available**
- If we assume that Process A completes, it will turn over its currently allocated resources, incrementing the Available vector.

  (2, 1, 2, 1) Current value of **Available**
  \+ (2, 0, 1, 1) **Allocation** (Process A)
  ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `

  (4, 1, 3, 2) Updated value of **Available**

Iteration 3:

- Examine the **Need** matrix without the row for Process D and Process A. The only row that is less than the Available vector is the one for Process B.
  Need(Process B) = (0, 1, 1, 2) < (4, 1, 3, 2) = **Available**
- If we assume that Process B completes, it will turn over its currently allocated resources, incrementing the Available vector.

  (4, 1, 3, 2) Current value of **Available**
  \+ (0, 1, 0, 0) **Allocation** (Process B)
  ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` ` `

  (4, 2, 3, 2) Updated value of **Available**

Iteration 4:

- Examine the **Need** matrix without the rows for Process A, Process B, and Process D. The only row left is the one for Process C, and it is less than the **Available** vector.

  Need(Process C) = (3, 1, 0, 0) < (4, 2, 3, 2) = **Available**
- If we assume that Process C completes, it will turn over its currently allocated resources, incrementing the **Available** vector.

(4, 2, 3, 3)	Current value of **Available**
+ (1, 0, 1, 1)	**Allocation** (Process C)
\` \` \` \` \` \` \` \` \` \` \` \` \` \` \` \` \` \`	
(5, 2, 4, 3)	Updated value of **Available**

Notice that the final value of the Available vector is the same as the original Total vector, showing the total number of all resources:

Total = (5, 2, 4, 2) < (5, 2, 4, 2) = Available

This means that the initial state represented by the Allocation and Need matrices is a safe state. The safe sequence that assures this safe state is **<D, A, B, C>.**

Problem: FInd the safe sequence for the following allocation

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	1	1	0	0	2	1	0	1	5	2	0
P_1	1	2	3	1	1	6	5	2				
P_2	1	3	6	5	2	3	6	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				