****

**Student Name:** Md Asibur Rahman Akash

**Student ID:** 11800427

**Section:** K18PT

**Email Address:** text2mine@gmail.com

**GitHub Link:** <https://github.com/Aakash-17>

**Question:24** There are 5 processes and 3 resource types, resource A with 10 instances, B with 5 instances and C with 7 instances. Consider following and write a c code to find whether the system is in safe state or not?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Available | | | Processes | Allocation | | | Max | | |
| A | B | C | A | B | C | A | B | C |
| 3 | 3 | 2 | P0 | 0 | 1 | 0 | 7 | 5 | 3 |
|  | | | P1 | 2 | 0 | 0 | 3 | 2 | 2 |
| P2 | 3 | 0 | 2 | 9 | 0 | 2 |
| P3 | 2 | 1 | 1 | 2 | 2 | 2 |
| P4 | 0 | 0 | 2 | 4 | 3 | 3 |

***Description:***

The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue.

Let **‘n’**be the number of processes in the system and **‘m’**be the number of resources types.

**Available :**

* It is a 1-d array of size **‘m’** indicating the number of available resources of each type.
* Available[ i ] = k means there are **‘k’** instances of resource type **Rj**

**Max:**

* It is a 2-d array of size ‘**n\*m’**that defines the maximum demand of each process in a system.
* Max [ i , j] = k means process **Pi** may request at most **‘k’** instances of resource type **Rj.**

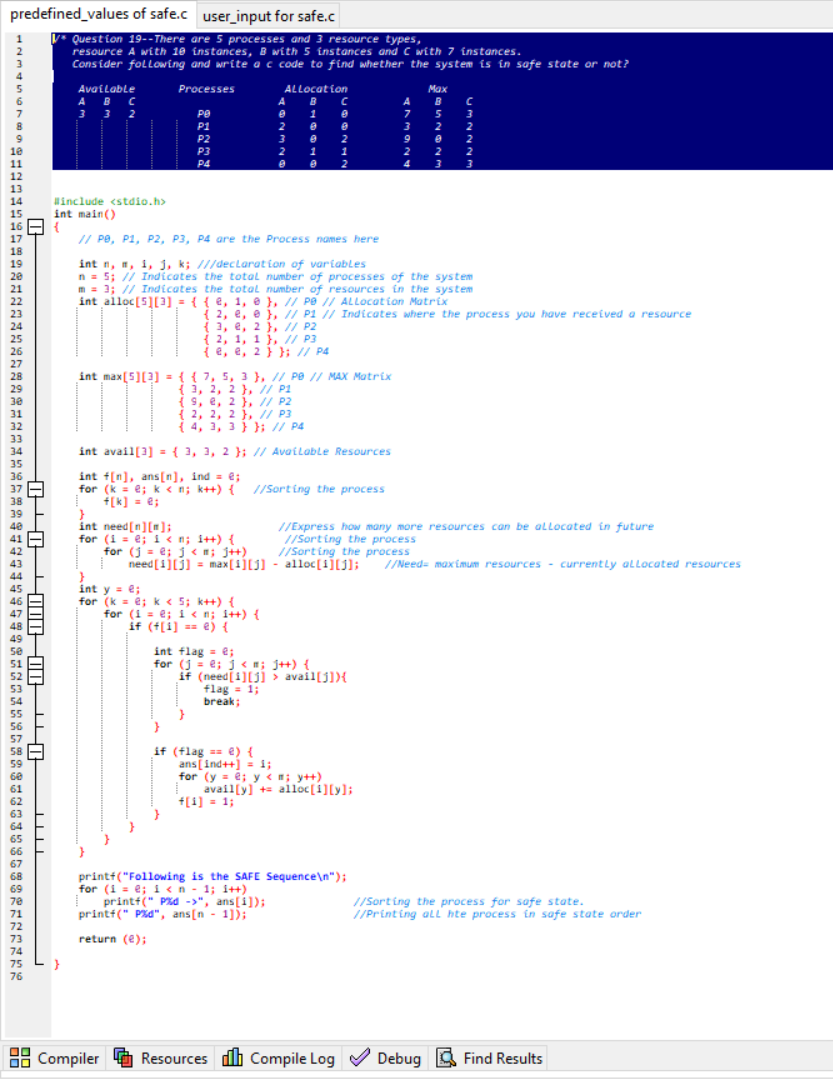
**Allocation:**

* It is a 2-d array of size**‘n\*m’**that defines the number of resources of each type currently allocated to each process.
* Allocation [ I , j] = k means process **Pi** is currently allocated **‘k’** instances of resource type **Rj**

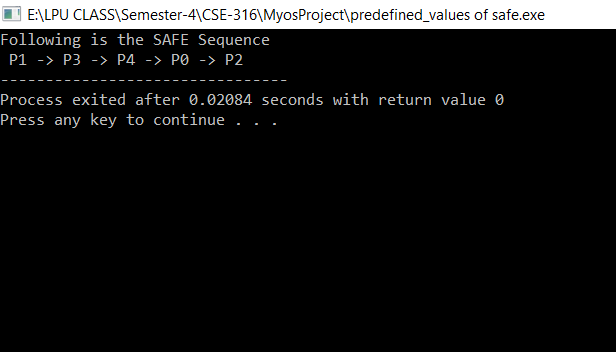
**Need :**

* It is a 2-d array of size **‘n\*m’** that indicates the remaining resource need of each process.
* Need [ i ,  j ] = k means process **Pi** currently need **‘k’** instances of resource type **Rj**

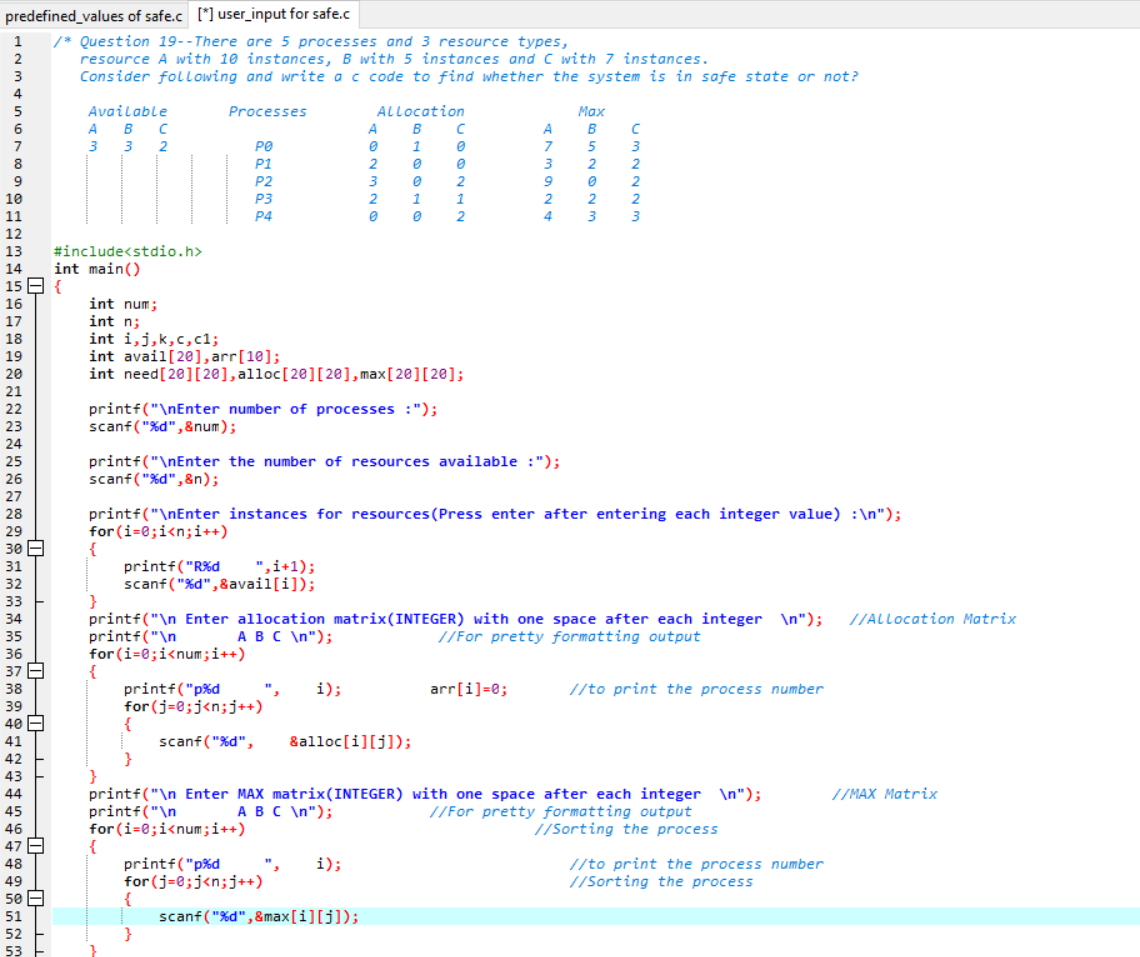
for its execution.

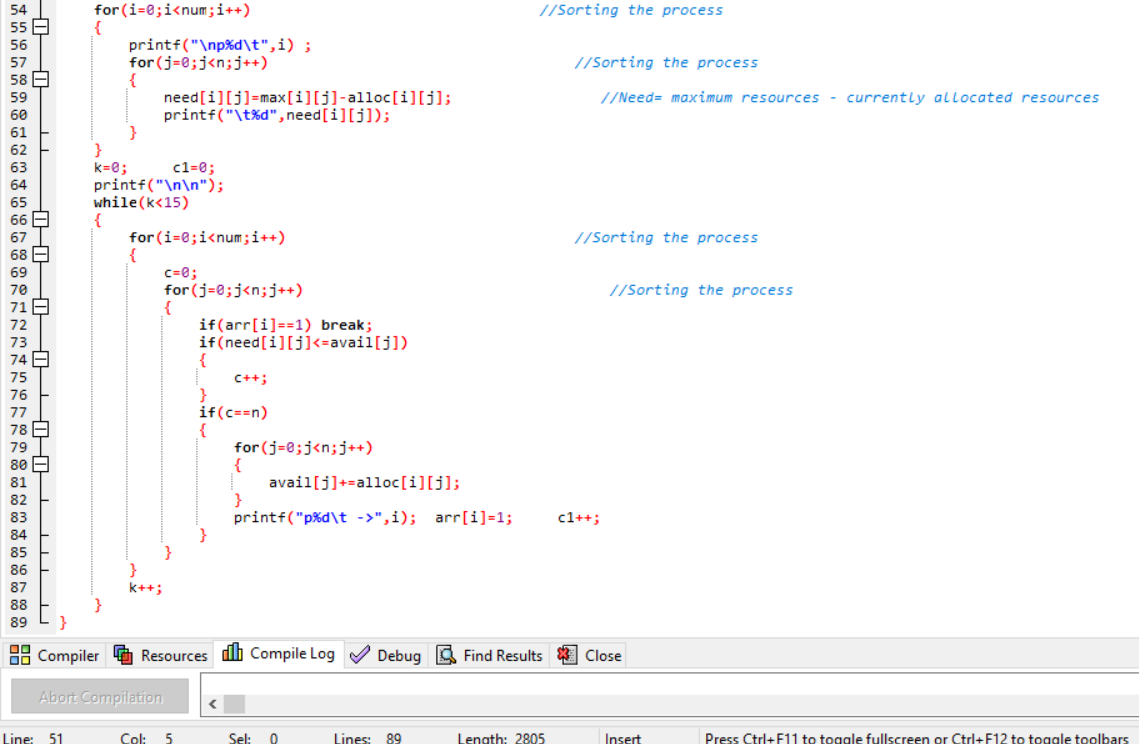
**Code:-(Predefined values in the code**

**Output:**

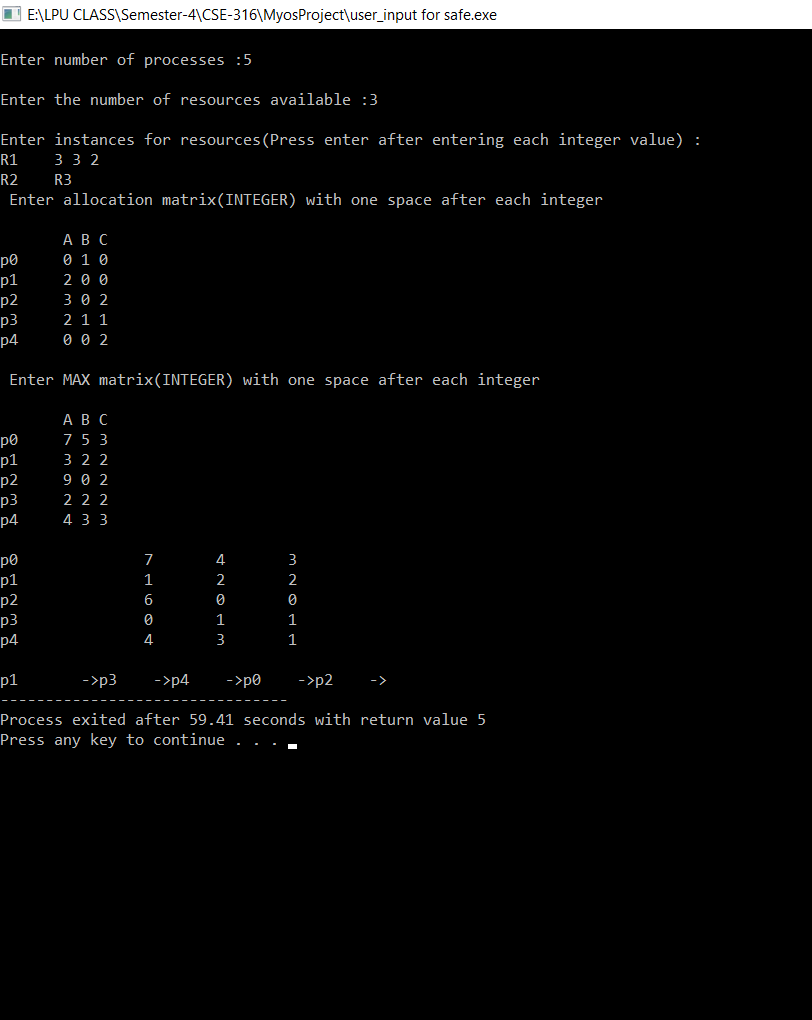


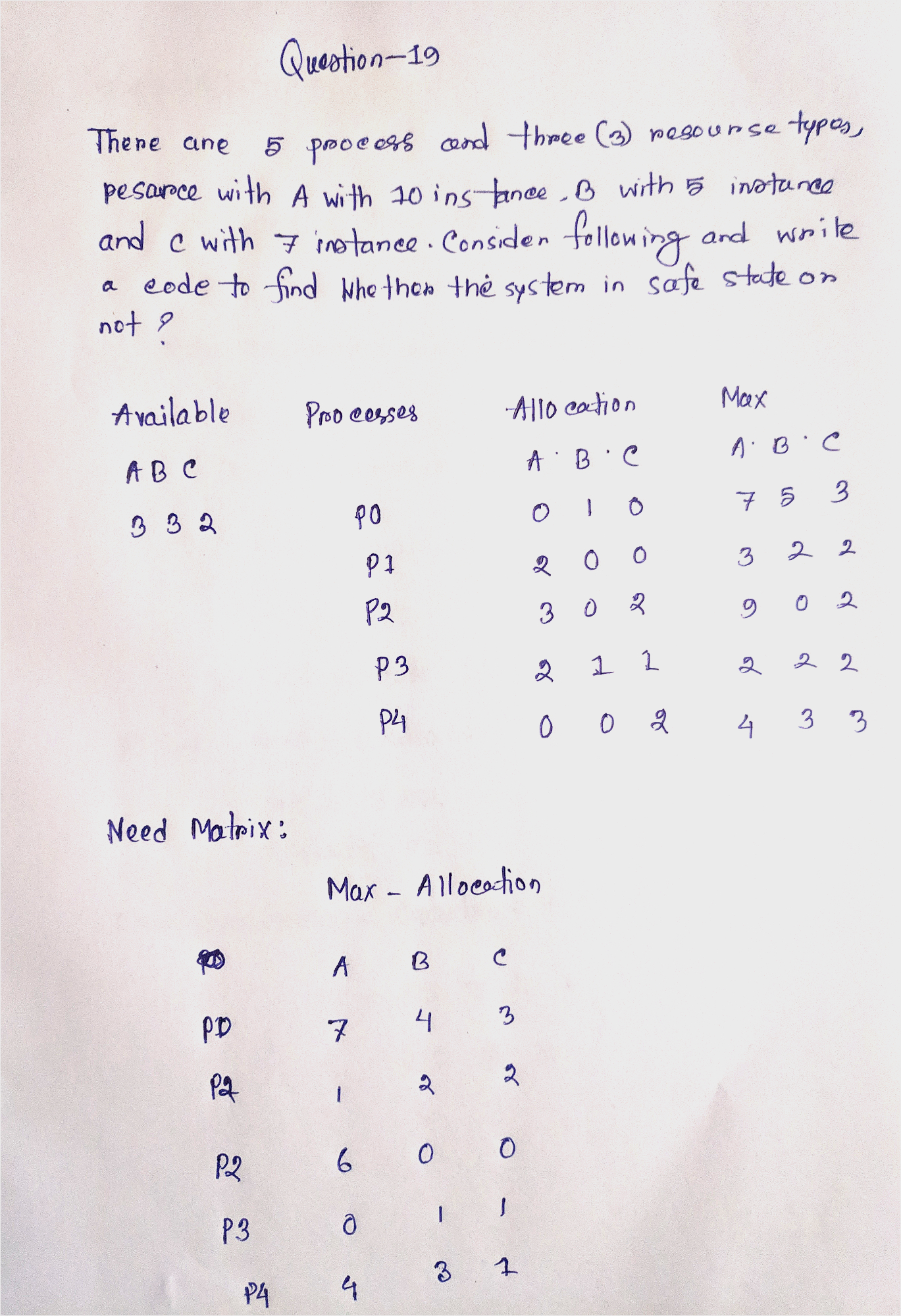
**Code:-(User is asked to enter the values)**

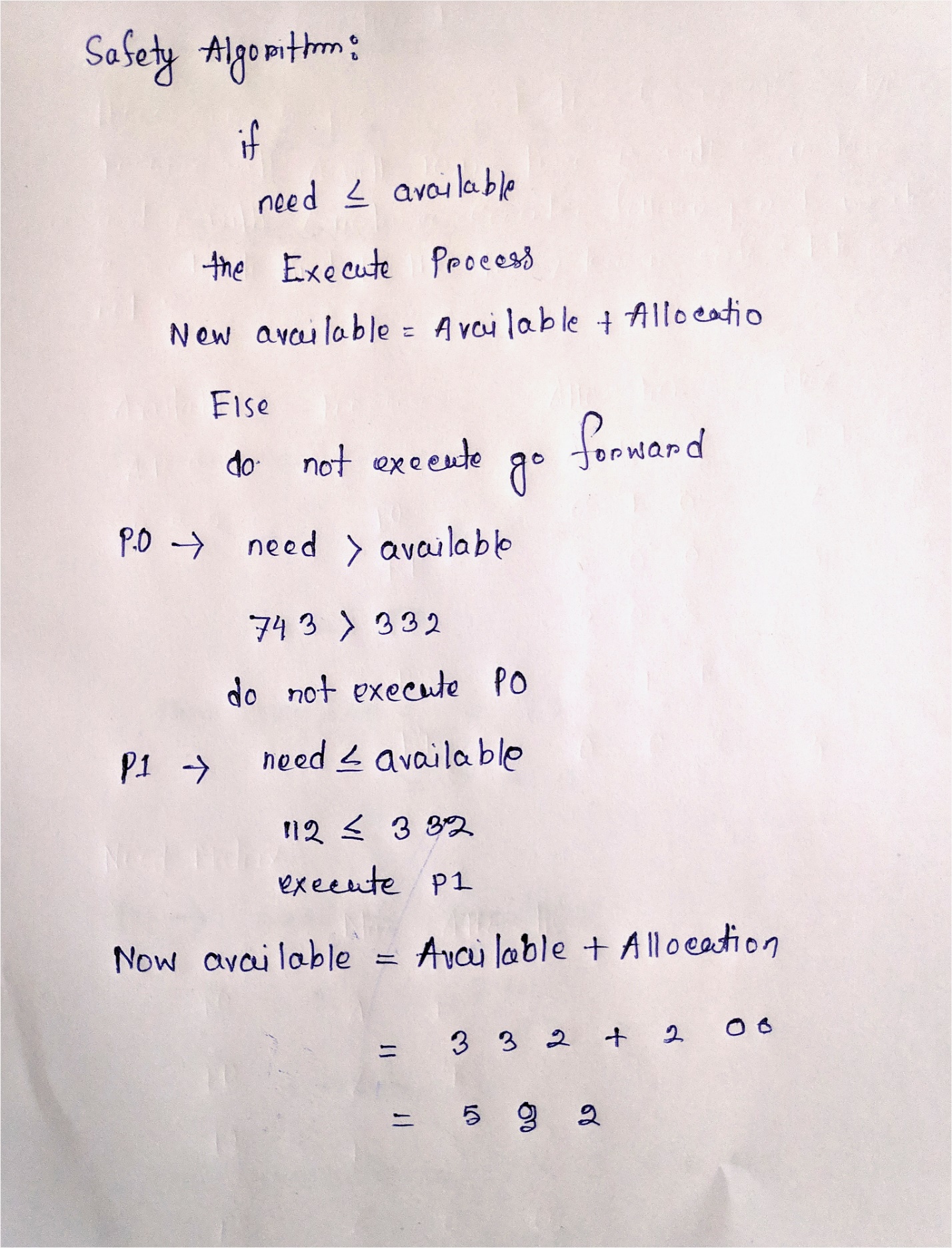
****

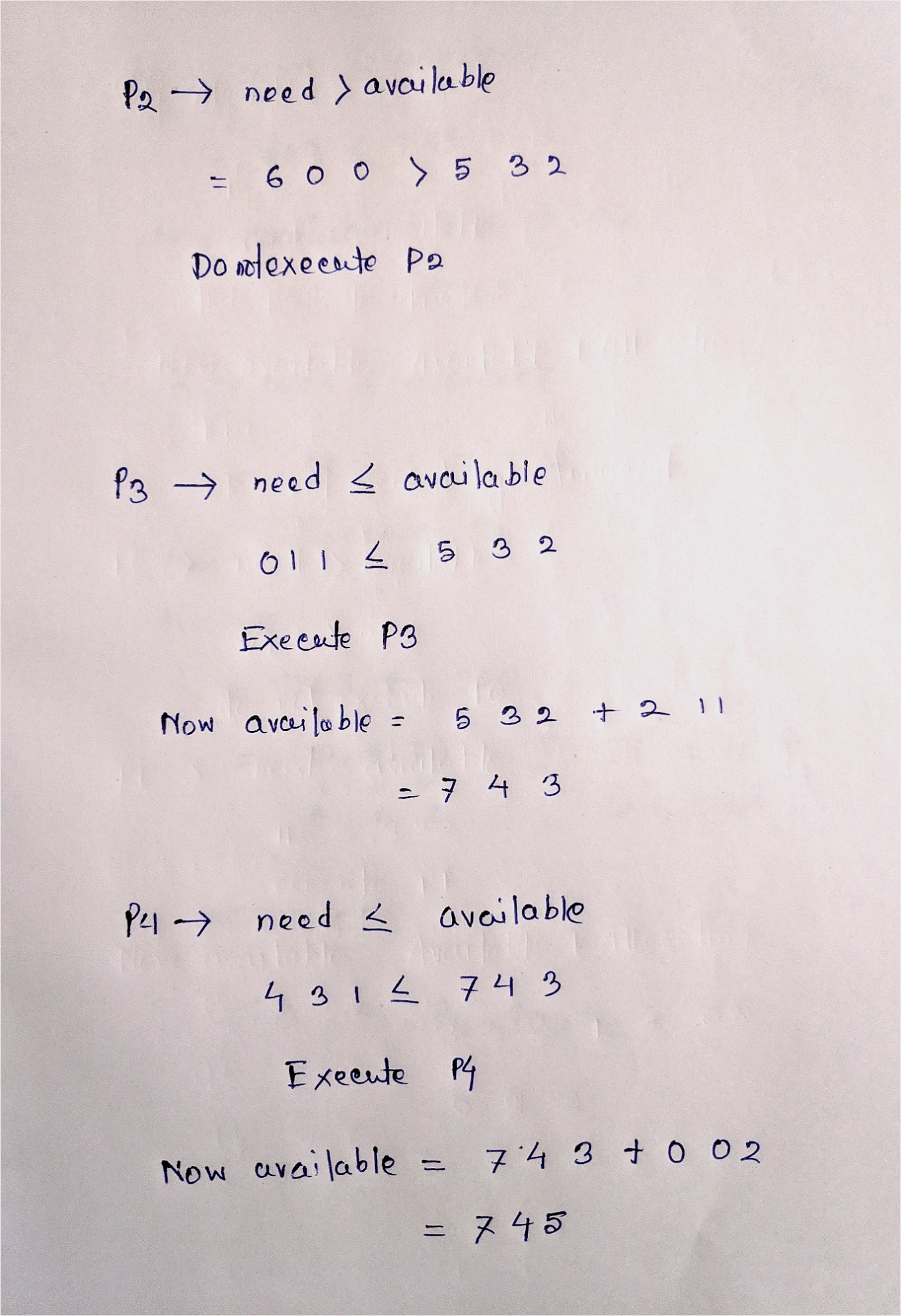
****

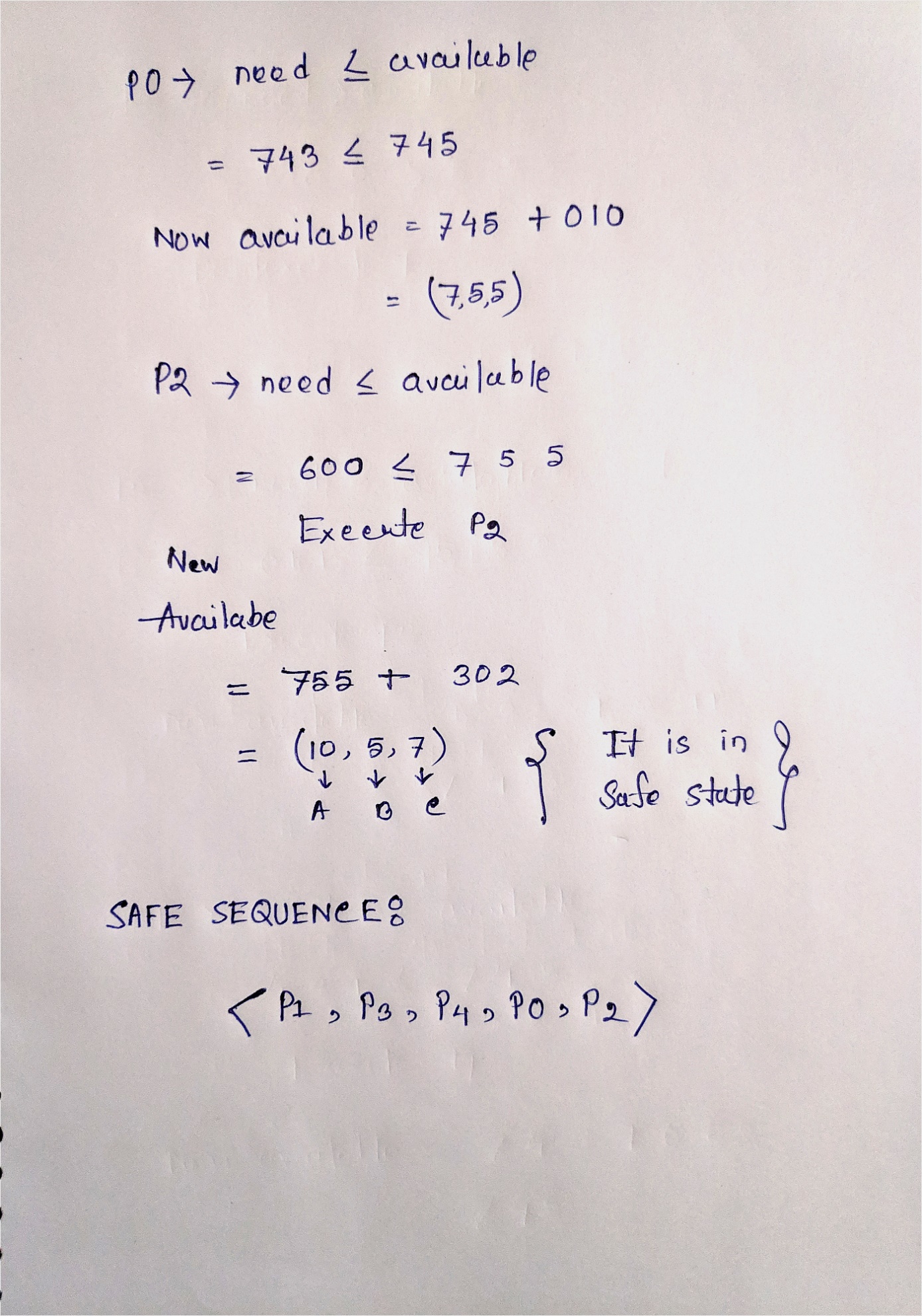
**Output:**

****

****

****

****

****