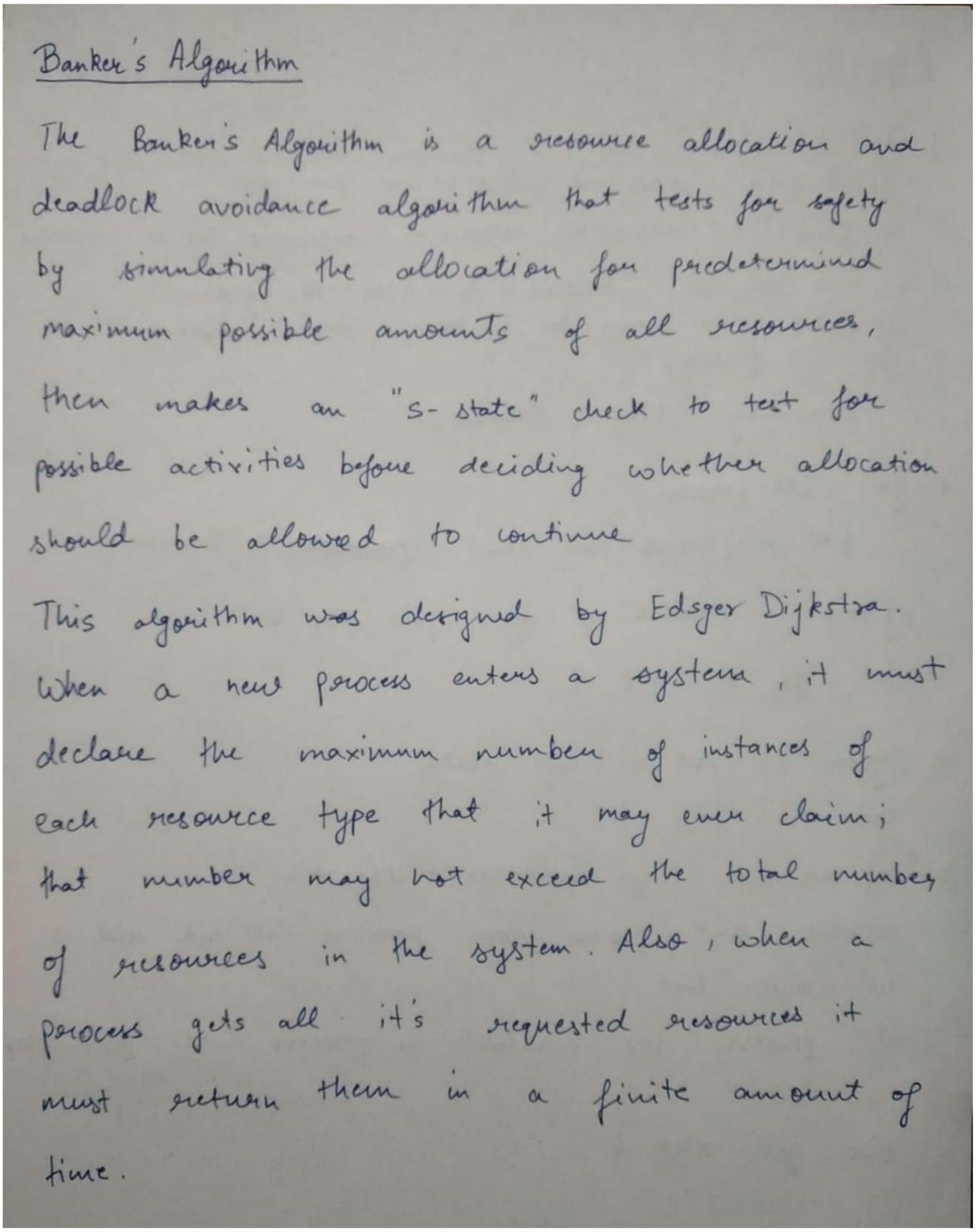
Experiment – 7

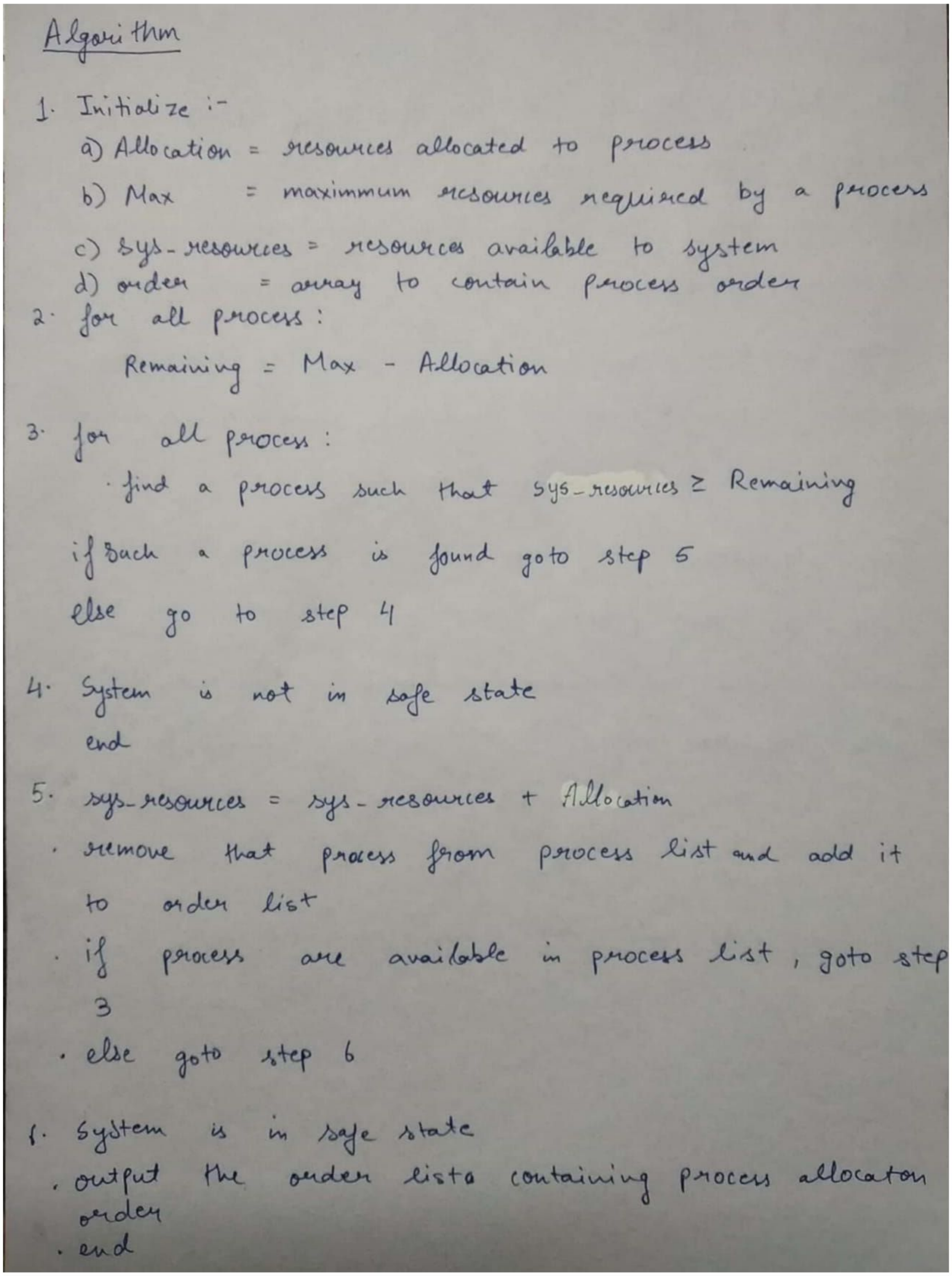


Banker's Algorithm

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

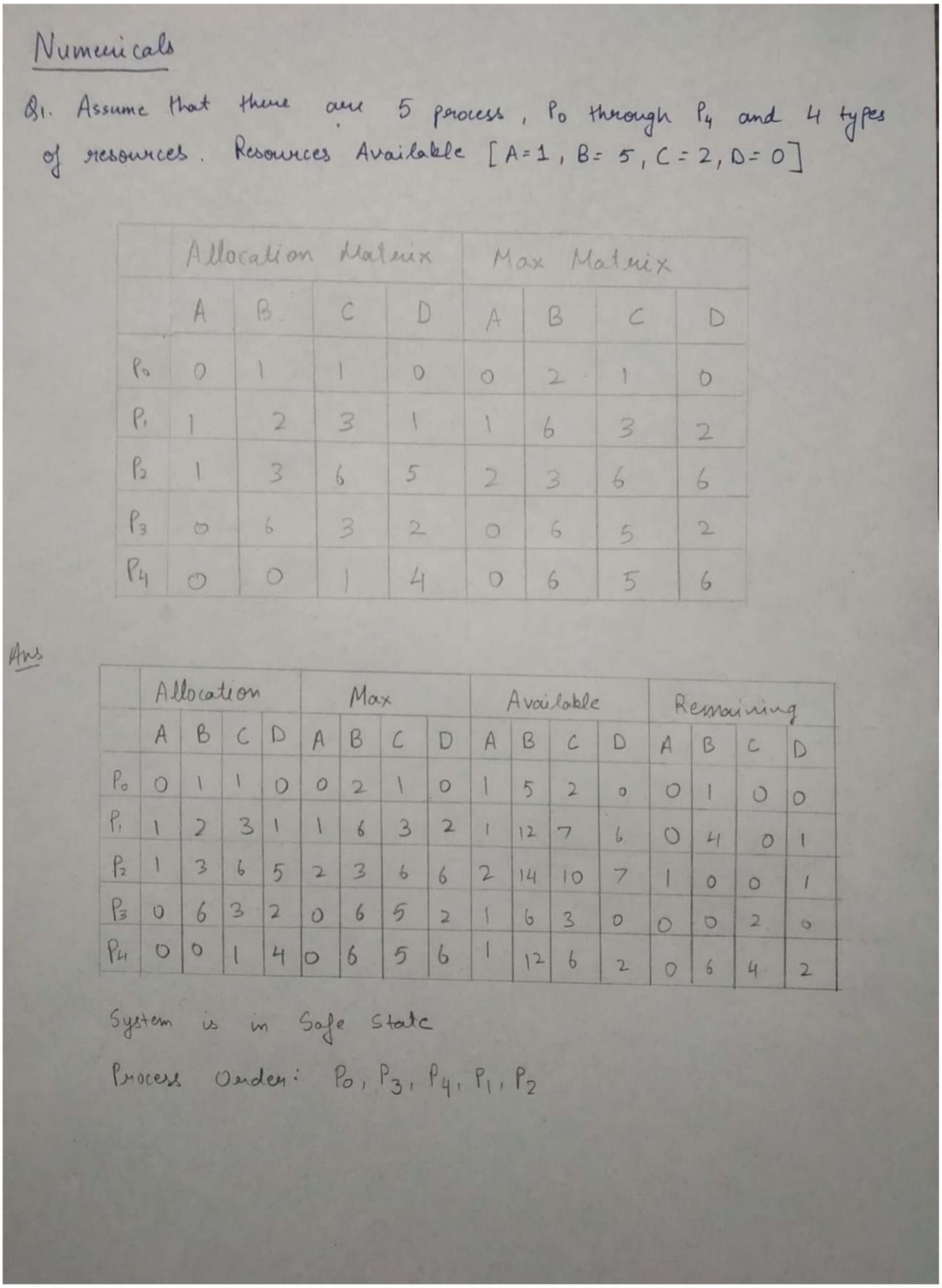
This algorithm was designed by Edsger Dijkstra.

When a new process enters a system, it must declare the maximum number of instances of each resource type that it may ever claim; that number may not exceed the total number of resources in the system. Also, when a process gets all it's requested resources it must return them in a finite amount of time.



Algorithm

```
1. Initialize :-
    a) Allocation = resources allocated to process
    b) Max = maximmum resources required by a process
    c) sys-resources = resources available to system
    d) order = array to contain process order
2. for all process :
       Remaining = Max - Allocation

3. for all process :
      · find a process such that sys-resources ≥ Remaining

   if such a process is found goto step 5
   else go to step 4

4. System is not in safe state
   end

5. sys-resources = sys-resources + Allocation
   · remove that process from process list and add it
     to order list
   · if process are available in process list , goto step
     3
   · else goto step 6

6. System is in safe state
   · output the order lista containing process allocaton
     order
   · end
```



Numericals

Q1. Assume that there are 5 process, P_0 through P_4 and 4 types of resources. Resources Available [A=1, B=5, C=2, D=0]

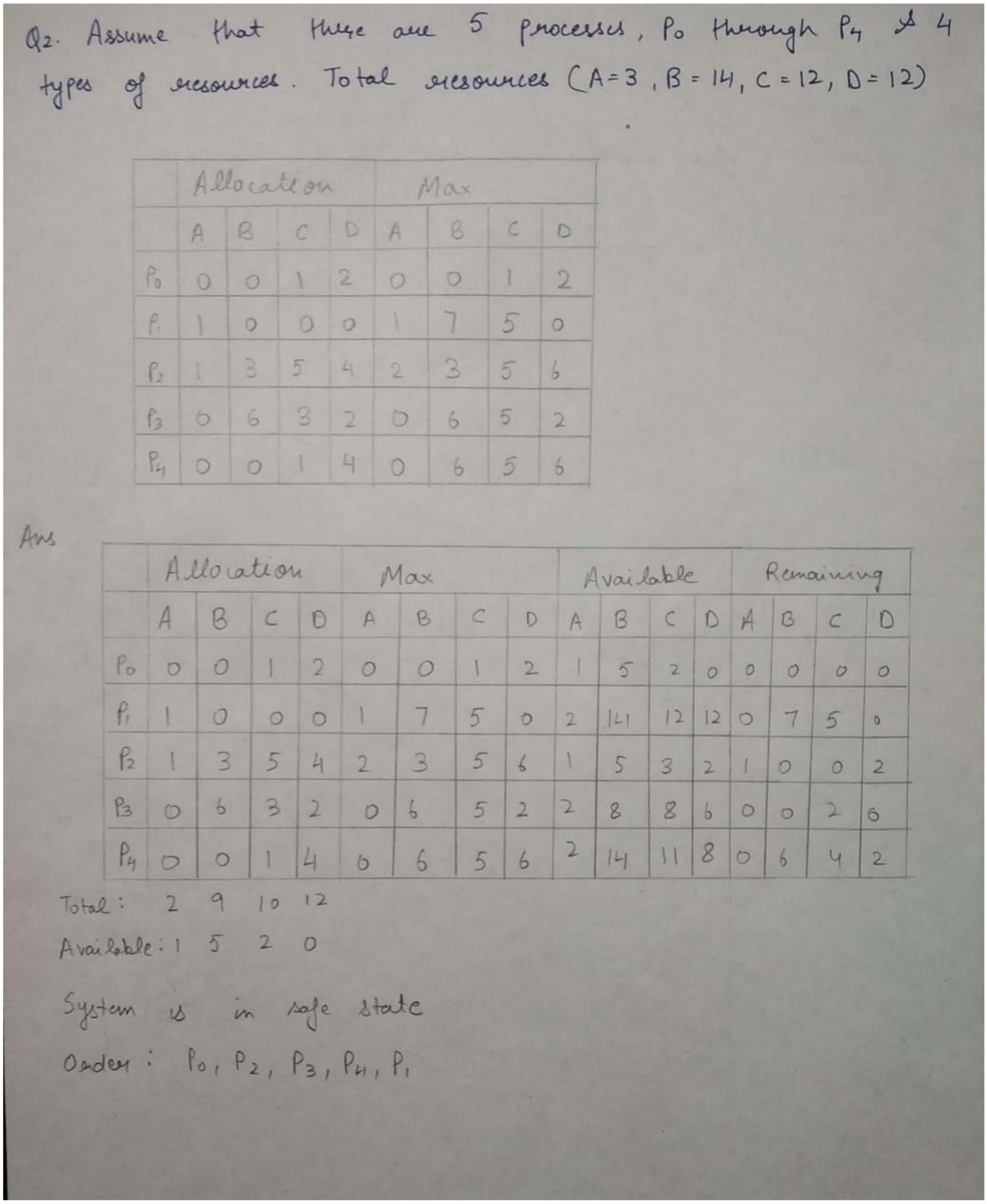
	Allocation Matrix				Max Matrix			
	A	B	C	D	A	B	C	D
P_0	0	1	1	0	0	2	1	0
P_1	1	2	3	1	1	6	3	2
P_2	1	3	6	5	2	3	6	6
P_3	0	6	3	2	0	6	5	2
P_4	0	0	1	4	0	6	5	6

Ans

	Allocation				Max				Available				Remaining			
	A	B	C	D	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	0	1	0	0
P_1	1	2	3	1	1	6	3	2	1	12	7	6	0	4	0	1
P_2	1	3	6	5	2	3	6	6	2	14	10	7	1	0	0	1
P_3	0	6	3	2	0	6	5	2	1	6	3	0	0	0	2	0
P_4	0	0	1	4	0	6	5	6	1	12	6	2	0	6	4	2

System is in Safe State

Process Order: P_0, P_3, P_4, P_1, P_2



Q2. Assume that there are 5 processes, P_0 through P_4 & 4 types of resources. Total resources (A = 3, B = 14, C = 12, D = 12)

	Allocation				Max			
	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2
P_1	1	0	0	0	1	7	5	0
P_2	1	3	5	4	2	3	5	6
P_3	0	6	3	2	0	6	5	2
P_4	0	0	1	4	0	6	5	6

Ans

	Allocation				Max				Available				Remaining			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0	0	0	0	0
P_1	1	0	0	0	1	7	5	0	2	14	12	12	0	7	5	0
P_2	1	3	5	4	2	3	5	6	1	5	3	2	1	0	0	2
P_3	0	6	3	2	0	6	5	2	2	8	8	6	0	0	2	0
P_4	0	0	1	4	0	6	5	6	2	14	11	8	0	6	4	2

Total: 2 9 10 12

Available: 1 5 2 0

System is in safe state

Order: P_0, P_2, P_3, P_4, P_1

Code

```
In [1]: import pandas as pd
        import numpy as np
        import string
```

```
In [2]: no_resources = int(input("Number of resources: "))
        resources = [string.ascii_uppercase[x] for x in range(no_resources)]
        cols = pd.MultiIndex.from_product([
            ['Allocation','Max','Available','Remaining'],
            resources
        ])

        process_count = int(input("Number Of Processes: "))
        index = [f'P{x+1}' for x in range(process_count)]

        df = pd.DataFrame(index=index, columns=cols, dtype=np.int64)
```

```
Number of resources: 4
Number Of Processes: 5
```

```
In [3]: for x in ['Allocation','Max']:
            for y in resources:
                df.loc[:,(x,y)] = list(map(int, input(f"{x} {y} : ").split()))
```

```
Allocation A : 0 1 1 0 0
Allocation B : 1 2 3 6 0
Allocation C : 1 3 6 3 1
Allocation D : 0 1 5 2 4
Max A : 0 1 2 0 0
Max B : 2 6 3 6 6
Max C : 1 3 6 5 5
Max D : 0 2 6 2 6
```

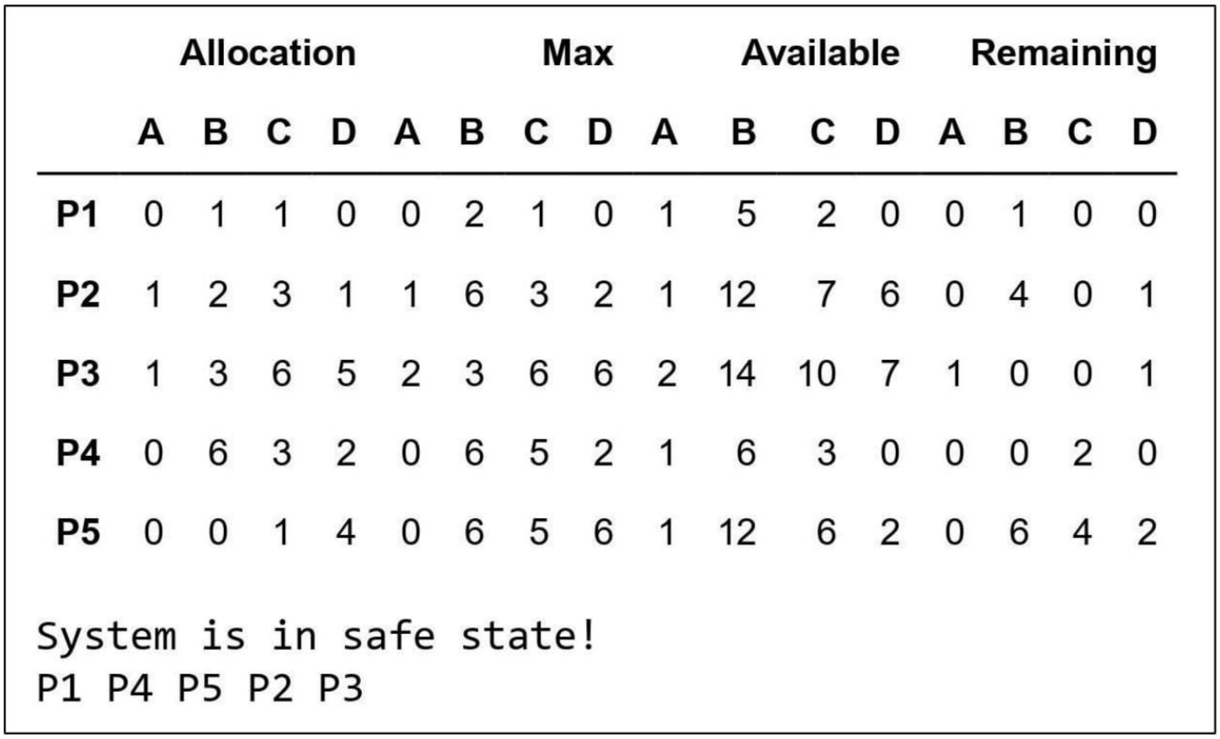
```
In [4]: df['Remaining'] = df['Max'] - df['Allocation']
        df['Available'] = 0
```

```
In [5]: sys_resourcces = np.array(list(map(
            int,
            input("Free Resources: ").split()
        ))).astype(np.int64)
```

```
Free Resources: 1 5 2 0
```

```
In [6]: order = []
        found = 1
        while len(order)<process_count:
            if not found:
                break
            found = 0
            for index,row in df.iterrows():
                if index not in order:
                    if(np.all(sys_resourcces>=df['Remaining'].loc[index])):
                        order.append(index)
                        df.loc[index,'Available'] = sys_resourcces.tolist()
                        sys_resourcces += df['Allocation'].loc[index]
                        found=1

        df['Available'] = df['Available'].astype(np.int64)
        display(df)
        if not found:
            print("System is not in safe state!")
        else:
            print("System is in safe state!")
            print(*order)
```

Output



	Allocation				Max				Available				Remaining			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P1	0	1	1	0	0	2	1	0	1	5	2	0	0	1	0	0
P2	1	2	3	1	1	6	3	2	1	12	7	6	0	4	0	1
P3	1	3	6	5	2	3	6	6	2	14	10	7	1	0	0	1
P4	0	6	3	2	0	6	5	2	1	6	3	0	0	0	2	0
P5	0	0	1	4	0	6	5	6	1	12	6	2	0	6	4	2

```
System is in safe state!
P1 P4 P5 P2 P3
```