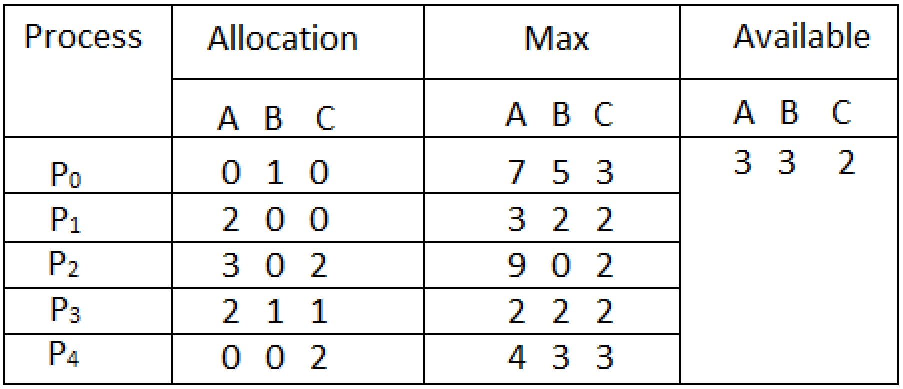
WEEK 7

Yash Gupta
1BM21CS251
26-07-2023

Q: Use bankers algo given here to check if the following state is safe/unsafe:

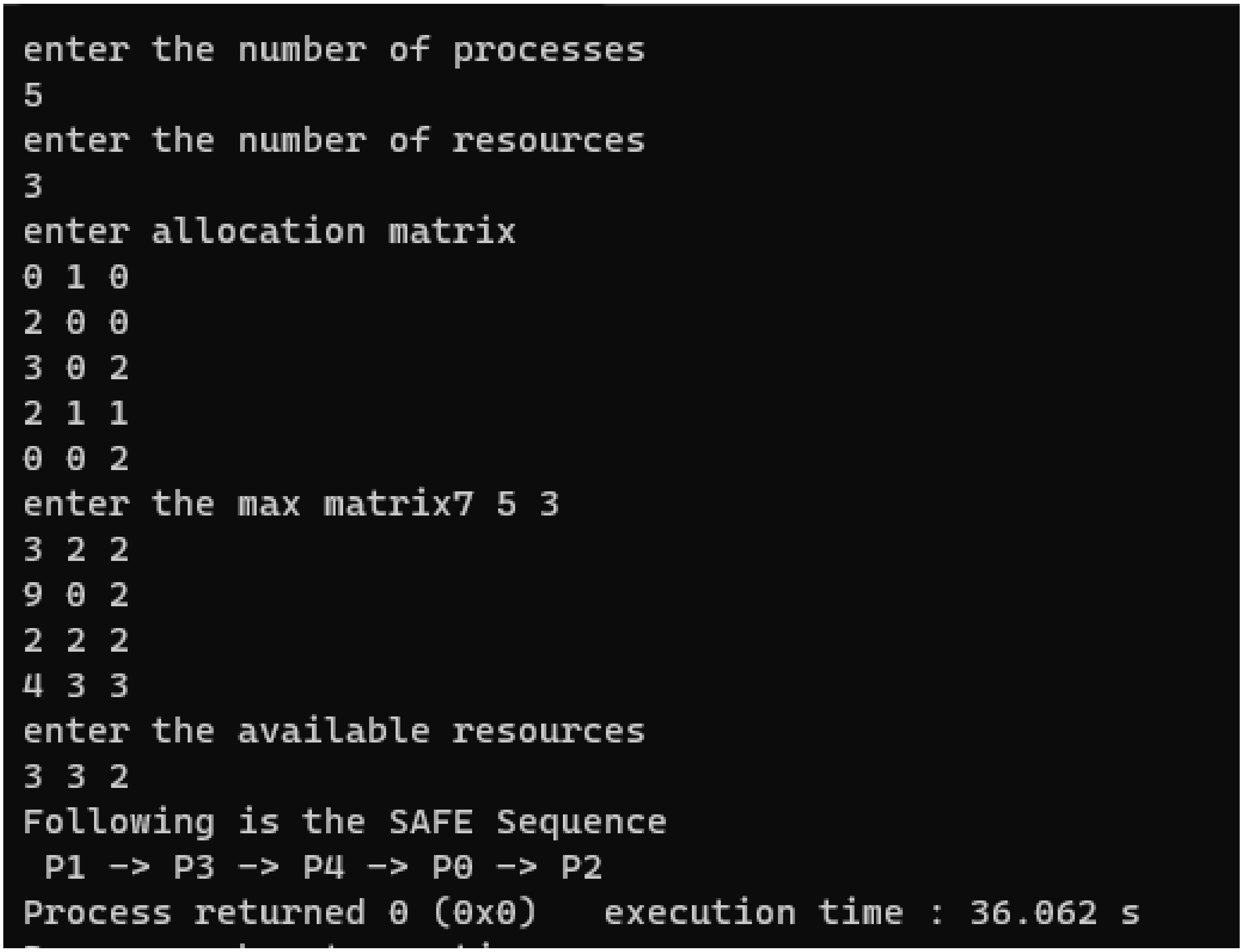


Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

```
#include <stdio.h>
int main()
{
    int n, m, i, j, k;
    printf("enter the number of processes\n");
    scanf("%d", &n);
    printf("enter the number of resources\n");
    scanf("%d", &m);
    int alloc[n][m], max[n][m];
    printf("enter allocation matrix\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < m; j++)
    {
        scanf("%d", &alloc[i][j]);
    }
}
printf("enter the max matrix");
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
    {
        scanf("%d", &max[i][j]);
    }
}
int avail[m];
printf("enter the available resources\n");
for (i = 0; i < m; i++)
{
    scanf("%d", &avail[i]);
}
int f[n], ans[n], ind = 0;
for (k = 0; k < n; k++)
{
    f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
        need[i][j] = max[i][j] - alloc[i][j];
}
int y = 0;
for (k = 0; k < 5; k++)
{
    for (i = 0; i < n; i++)
    {
        if (f[i] == 0)
        {
            int flag = 0;
            for (j = 0; j < m; j++)
            {
                if (need[i][j] > avail[j])
                {
                    flag = 1;
                    break;
                }
            }
            if (flag == 0)
            {
                ans[ind++] = i;
                for (y = 0; y < m; y++)
                    avail[y] += alloc[i][y];
                f[i] = 1;
            }
        }
    }
}
int flag = 1;
for (int i = 0; i < n; i++)
{
    if (f[i] == 0)
    {
        flag = 0;
        printf("The following system is not safe");
        break;
    }
}
if (flag == 1)
```

```
    {
      printf("Following is the SAFE Sequence\n");
      for (i = 0; i < n - 1; i++)
        printf(" P%d ->", ans[i]);
      printf(" P%d", ans[n - 1]);
    }
    return (0);
}
```

OUTPUT:



```
enter the number of processes
5
enter the number of resources
3
enter allocation matrix
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
enter the max matrix7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
enter the available resources
3 3 2
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2
Process returned 0 (0x0)   execution time : 36.062 s
```