```
// ////////////////////////////////////////////////////////Banker's Algorithm
#include <stdio.h>
int main()
{
    // P0, P1, P2, P3, P4 are the Process names here
  int avail[10], alloc[10][10], max[10][10], need[10][10], maxres[10], m, n, i,j,k,sum;

    printf("\nEnter the number of processes and the number of resources:\n");
	scanf("%d%d", &n, &m);

	printf("\nEnter maximum instances of resources\n");
	for (j = 0; j < m; j++)
	{
		scanf("%d", &maxres[j]);
		avail[j] = maxres[j];
	}

	printf("\nEnter the Allocated Matrix:\n");
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
			scanf("%d", &alloc[i][j]);
	}

	printf("\nEnter the Max Matrix:\n");
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
		{
			scanf("%d", &max[i][j]);
			need[i][j] = max[i][j] - alloc[i][j];
		}
	}

	printf("\nThe Need Matrix is:\n");
	for (i = 0; i < n; i++)
	{
		for (j = 0; j < m; j++)
			printf("%d ", need[i][j]);
		printf("\n");
	}

	for (j = 0; j < m; j++)    //calculating available matrix after allocation
	{
		sum = 0;
		for (i = 0; i < n; i++)
			sum += alloc[i][j];
		avail[j] -= sum;
	}
```

```
    int finish[10],safeseq[10], ind = 0;
    for (k = 0; k < n; k++) {
        finish[k] = 0;
    }



    int y = 0;
    for (k = 0; k < n; k++) {
        for (i = 0; i < n; i++) {
            if (finish[i] == 0)
            {

                int flag = 0;
                for (j = 0; j < m; j++)
                {
                    if (need[i][j] > avail[j])
                    {
                        flag = 1;
                         break;
                    }
                }

                if (flag == 0)
                {
                    safeseq[ind++] = i;
                    for (y = 0; y < m; y++)
                        avail[y] += alloc[i][y];
                    finish[i] = 1;
                    //printf("i=%d\n",i);
                }
            }
        }
    }

    for (i = 0; i < n; i++)
      if (finish[i] == 0)
            {
            printf("system is in unsafe state.");
            return(0);
                }


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

}
```

```
/* OUTPUT:(try with some unsafe state also)


Enter the number of processes and the number of resources:
5
3

Enter maximum instances of resources
10
5
7

Enter the Allocated Matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter the Max Matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

The Need Matrix is:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
i=1
i=3
i=4
i=0
i=2
Following is the SAFE Sequence
 P1 -> P3 -> P4 -> P0 -> P2

*/
```