int main() {  
    int n, m; // n = number of processes, m = number of resources  
    printf("Enter the number of processes: ");  
    scanf("%d", &n);  
    printf("Enter the number of resources: ");  
    scanf("%d", &m);  
  
    int max[n][m];       // Maximum resource needs for each process  
    int allocated[n][m]; // Resources currently allocated to each process  
    int need[n][m];      // Remaining resource needs for each process  
    int available[m];    // Available resources  
  
    // Input maximum resource needs for each process  
    printf("Enter the maximum resource needs for each process:\n");  
    for (int i = 0; i < n; i++) {  
        printf("Process %d: ", i);  
        for (int j = 0; j < m; j++) {  
            scanf("%d", &max[i][j]);  
        }  
    }  
  
    // Input resources currently allocated to each process  
    printf("Enter the resources currently allocated to each process:\n");  
    for (int i = 0; i < n; i++) {  
        printf("Process %d: ", i);  
        for (int j = 0; j < m; j++) {  
            scanf("%d", &allocated[i][j]);  
        }  
    }  
  
    // Input available resources  
    printf("Enter the available resources: ");  
    for (int i = 0; i < m; i++) {  
        scanf("%d", &available[i]);  
    }  
  
    // Calculate remaining resource needs for each process  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < m; j++) {  
            need[i][j] = max[i][j] - allocated[i][j];  
        }  
    }  
  
    // Initialize a boolean array to track whether each process can be satisfied  
    int finish[n];  
    for (int i = 0; i < n; i++) {  
        finish[i] = 0; // Initially, no processes are finished  
    }  
  
    int sequence[n]; // To store the safe execution sequence  
    int count = 0;   // Count of finished processes  
  
    // Loop to find a safe sequence  
    while (count < n) {  
        int found = 0; // Flag to check if a process can be satisfied in this iteration  
  
        // Try to find a process that can be satisfied  
        for (int i = 0; i < n; i++) {  
            if (finish[i] == 0) {  
                int j;  
                for (j = 0; j < m; j++) {  
                    if (need[i][j] > available[j]) {  
                        break;  
                    }  
                }  
                if (j == m) {  
                    // Process i can be satisfied  
                    for (int k = 0; k < m; k++) {  
                        available[k] += allocated[i][k];  
                      printf("%d\t\n",available[k]);  
                    }  
                    sequence[count++] = i;  
                    finish[i] = 1;  
                    found = 1;  
                }  
            }  
        }  
         
       
  
        if (!found) {  
            // If no process can be satisfied in this iteration, the system is not in a safe state  
            printf("The system is in an unsafe state!\n");  
            return 0;  
        }  
    }  
      need[n][m]=max[n][m]-allocated[n][m];  
      printf("need matrix:\n");  
        for(int i=0;i<n;i++)  
        {  
        printf("\n");  
        for(int j=0;j<m;j++)  
        {  
        printf("%d\t",need[i][j]);  
        }  
        }  
  
    // If we reach here, a safe sequence has been found  
    printf("Safe Sequence: \n");  
    for (int i = 0; i < n; i++) {  
        printf("P%d ", sequence[i]);  
    }  
    printf("\n");  
  
    return 0;  
}  
/\*output:  
  
comp54@comp54:~/Desktop$ gcc bankers.c  
comp54@comp54:~/Desktop$ ./a.out  
Enter the number of processes: 5    
Enter the number of resources: 3  
Enter the maximum resource needs for each process:  
Process 0: 4 3 3  
Process 1: 3 2 2  
Process 2: 9 0 2  
Process 3: 7 5 3  
Process 4: 1 1 2  
Enter the resources currently allocated to each process:  
Process 0: 1 1 2  
Process 1: 2 1 2  
Process 2: 4 0 1  
Process 3: 0 2 0  
Process 4: 1 1 2  
Enter the available resources: 2 1 0  
4  
2  
2  
5  
3  
4  
6  
4  
6  
10  
4  
7  
10  
6  
7  
need matrix:  
  
3 2 1  
1 1 0  
5 0 1  
7 3 3  
0 0 0

Safe Sequence:  
P1 P4 P0 P2 P3\*/