



Banker

PRUTCHAKORN



Input

```
numberOfProcess = int(input())
numberOfResource = int(input())
Allocation = []
for i in range(0, numberOfProcess):
    values = [int(x) for x in input().split()]
    Allocation.append(values)

Maximum = []
for i in range(0, numberOfProcess):
    values = [int(x) for x in input().split()]
    Maximum.append(values)

Available = [int(x) for x in input().split()]
```

Output

```
print("-----RESULT-----")
print("Allocation Matrix")
print(Matrix(Allocation), "\n")
print("Maximum Matrix")
print(Matrix(Maximum), "\n")
print("Need Matrix")
Need = get_need(Maximum, Allocation)
print(Matrix(Need), "\n")
print("Available Vector")
print(Available, "\n")
total = get_available(Need, Allocation, Available[:])
print("Total")
print(total, "\n")
print("Allocated")
Allocated = [int(total[x] - Available[x]) for x in range(len(Available))]
print(Allocated, "\n")
print("Safe Sequence")
print(safe_sequence)
```

Get Need

```
def get_need(Maximum, Allocation):  
    row_size = len(Maximum)  
    col_size = len(Maximum[0])  
    Need = []  
    for i in range(0, row_size):  
        values = []  
        for j in range(0, col_size):  
            values.append(Maximum[i][j] - Allocation[i][j])  
        Need.append(values)  
    return Need
```

Get available

```
def get_available(Need, Allocation, Available):  
    i = 1  
    while len(safe_sequence) < len(Need):  
        if not is_safe_sequence(safe_sequence, i) and is_fulfill(Need, Available, i):  
            for j in range(0, len(Available)):  
                Available[j] += Allocation[i][j]  
            safe_sequence.append(i)  
            print("Current available vector -> ", Available, "\n")  
  
            i += 1  
  
            if i > len(Need) - 1:  
                i = 0  
    return Available
```


Is fulfill between Need and Available

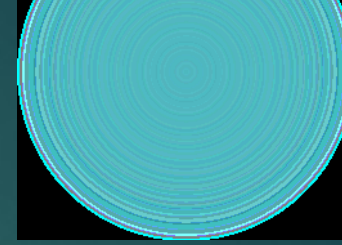
```
def is_fulfill(Need, Available, p):  
    size = len(Available)  
    count = 0  
    for i in range(0, size):  
        if Need[p][i] <= Available[i]:  
            count += 1  
    if count == size:  
        return True  
    else:  
        return False
```

Is safe sequence

```
def is_safe_sequence(safe_sequence, p):  
    for i in safe_sequence:  
        if i == p:  
            return True  
    return False
```



Show the matrix



```
def Matrix(A):  
    result = ""  
    for i in range(0, len(A)):  
        result += str(A[i]) + "\n"  
    return result
```



I/O

5		
3		
0	1	0
2	0	0
3	0	2
2	1	1
0	0	2
7	5	3
3	2	2
9	0	2
2	2	2
4	3	3
3	3	2

```
-----RESULT-----
Allocation Matrix
[0, 1, 0]
[2, 0, 0]
[3, 0, 2]
[2, 1, 1]
[0, 0, 2]

Maximum Matrix
[7, 5, 3]
[3, 2, 2]
[9, 0, 2]
[2, 2, 2]
[4, 3, 3]

Need Matrix
[7, 4, 3]
[1, 2, 2]
[6, 0, 0]
[0, 1, 1]
[4, 3, 1]

Available Vector
[3, 3, 2]
```

```
Current available vector -> [5, 3, 2]
Current available vector -> [7, 4, 3]
Current available vector -> [7, 4, 5]
Current available vector -> [7, 5, 5]
Current available vector -> [10, 5, 7]

Total
[10, 5, 7]

Allocated
[7, 2, 5]

Safe Sequence
[1, 3, 4, 0, 2]
```