Banker’s algorithm is a common method for deadlock avoidance. Given a system with n threads and m types of resources, we can define the system state with the following matrices and array:

Allocation: An $n{\times}m$ matrix. $Allocation[i,j]{=}k$ means thread T_i is currently allocated k instances of resource $R_{j.}$

Need: An $n{\times}m$ matrix. $Need[i,j]{=}k$ means thread T_i needs k more instances of resource R_j to complete execution.

Available: A vector of length m. $Available[j]{=}k$ means there are k instances of resource type R_j available.

Checking whether a system is in safety state is essential for deadlock avoidance. This can be done by the safety algorithm (← means assignment, and = checks equality):

```
1.  Let $\boldsymbol{Work}$ and $\boldsymbol{Finish}$ be vectors of length $m$ and $n$, respectively.  Initialize:
    $\boldsymbol{Work} \leftarrow \boldsymbol{Available}$
    $\boldsymbol{Finish}\ [\boldsymbol{i}] \leftarrow \boldsymbol{false}$ for $\boldsymbol{i} = \mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{n} - \mathbf{1}$
2.  Find an thread $T_i$ such that:
    $\boldsymbol{Finish}\,[\boldsymbol{i}] = \boldsymbol{false}$
    $\boldsymbol{Need}[\boldsymbol{i},\boldsymbol{j}] \le \boldsymbol{Work}[\boldsymbol{j}]$ for $\boldsymbol{j} = \mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{m} - \mathbf{1}$
    If no such $T_i$ exists, go to step 4.
3.  $\boldsymbol{Work}[j] \leftarrow \boldsymbol{Work}[j] + \boldsymbol{Allocation}[i,j]$ for $\boldsymbol{j} = \mathbf{0}, \mathbf{1}, \ldots, \boldsymbol{m} - \mathbf{1}$
    $\boldsymbol{Finish}[\boldsymbol{i}] \leftarrow \boldsymbol{true}$
    go to step 2
4.  If $\boldsymbol{Finish}\ [\boldsymbol{i}] = \boldsymbol{true}$ for all $\boldsymbol{i}$, then the system is in a safe state.
```

Your goal is to write a program to check if a system is in safety state by providing a class Banker which contains:

1. Private data members for the system state. Use single pointer to represent vector and double pointer to represent matrix.
2. Overloaded stream insertion operator for inputting system state.
3. Appropriate constructor for initializing an object with dynamically allocated memory.
4. Appropriate destructor for reclaiming the allocated memory.
5. A member function safety to check if a system is in safe sate.
6. An overloaded parentheses () operator with one argument i: check the safety after thread T_i requesting half of its needed resources. If the attempt is failed (go into unsafe state), then store back to the original state.
7. An overloaded parentheses () operator with two arguments i and j: check the safety after thread T_i requesting half of its needed resources R_j. If the attempt is failed (go into unsafe state), then store back to the original state.
8. Get functions for the private data members.

Requirement: Use the sample main function to complete your program. Provide a class Banker satisfying all of the above conditions and separate the interface (Banker.h) and implementation (Banker.cpp).

Prohibited: C-style code.

Input

Each case starts by two integers n and m, followed by $2n + 1$ lines which in turn correspond to the Allocation, Need, and Available. The input ends with two zeros for n and m.

Output

For each case, first output the Allocation$[n/2, m/2]$, the Need$[n/2, m/2]$, and the Available$[m/2]$. Then check the safety of the system. Output “Unsafe state” if system is unsafe; otherwise, further check the safety after two further requests: (1) half of the needed resources are allocated to thread $n/2$, and (2) half of the needed $m/2^{th}$ resource are allocated to thread $n/2$. If both result in unsafe state, then report “Unsafe state”; else, report “Safe state”.

Sample Input

```
5 3
6 4 3
7 0 5
2 0 7
5 6 1
8 1 8
5 6 5
9 2 6
5 2 3
2 1 2
2 8 8
4 4 2
```

Sample Output

```
0 2 4
Safe state
```