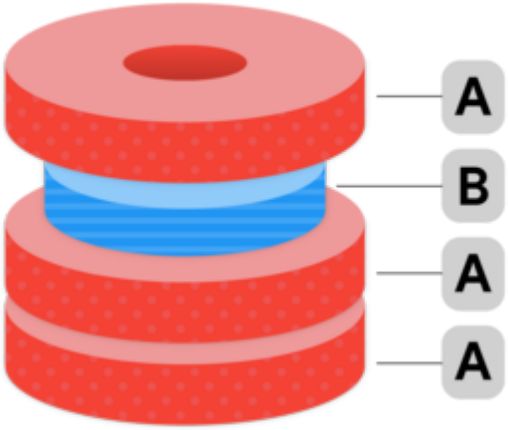
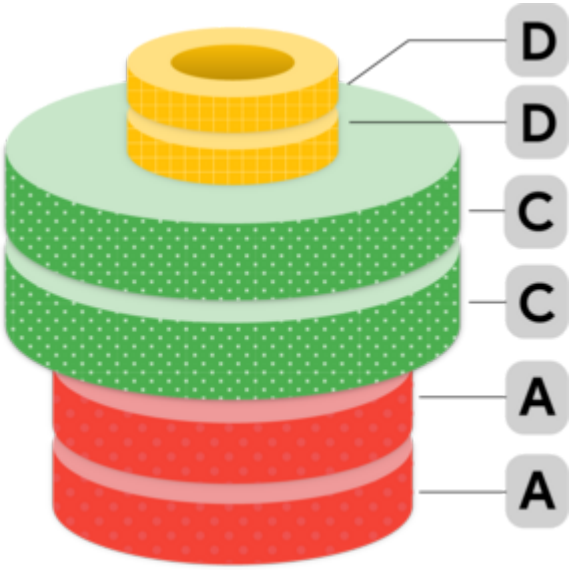
# Google Code Jam '22 Round 1A Problem C - Weightlifting

**Time limit:** 20.0s **Memory limit:** 1G

You are following a prescribed training for weightlifting. The training consists of a series of exercises that you must do in order. Each exercise requires a specific set of weights to be placed on a machine.

There are $W$ types of different weights. For example, an exercise may require $3$ weights of type A and $1$ weight of type B, while the next requires $2$ weights each of types A, C, and D.




The weights are placed on the machine as a stack. Formally, with a single operation, you can either add a new weight of any type to the top of the stack, or remove the weight that is currently at the top of the stack.

You can load the weights for each exercise onto the machine's stack in any order. So, if you place the weight of type B at the bottom in the first exercise of the example above, you will have to take all the weights off before putting on the weights for the second exercise. On the other hand, if you place the weight of type B third from the bottom, you can leave two of the weights of type A on the bottom of the stack to be part of the next exercise's set, saving you some time.

Given the amount of weights of each type needed for each exercise, find the minimum number of operations needed to do them all. You must complete the exercises in the order given. The machine stack starts out empty, and you must leave it empty after you finish with all your exercises.

## Input Specification

The first line of the input gives the number of test cases, $T$. $T$ test cases follow. Each test case starts with a line containing $2$ integers $E$ and $W$: the number of exercises and the number of types of weights. Weight types are numbered between $1$ and $W$. Then, $E$ lines follow. The $i^{\text{th}}$ of these lines contains $W$ integers $X_{i,1}, X_{i,2}, \ldots, X_{i,W}$ representing that the $i^{\text{th}}$ exercise requires exactly $X_{i,j}$ weights of type $j$.

## Output Specification

For each test case, output one line containing `Case #x: y`, where $x$ is the test case number (starting from 1) and $y$ is the minimum number of machine stack operations needed to run through all your exercises.

## Limits

Time limit: 20 seconds.

Memory limit: 1 GB.

$1 \leq T \leq 100$.

$1 \leq X_{i,1} + X_{i,2} + \ldots + X_{i,W}$, for all $i$. (Each exercise requires at least one weight.)

### Test Set 1

$1 \leq E \leq 10$.

$1 \leq W \leq 3$.

$0 \leq X_{i,j} \leq 3$, for all $i, j$.

### Test Set 2

$1 \leq E \leq 100$.

$1 \leq W \leq 100$.

$0 \leq X_{i,j} \leq 100$, for all $i, j$.

## Sample Input

```
3
3 1
1
2
1
2 3
1 2 1
2 1 2
3 3
3 1 1
3 3 3
2 3 3
```

## Sample Output

```
Case #1: 4
Case #2: 12
Case #3: 20
```

In Sample Case #1, there is only one type of weight. The first exercise needs $1$ weight, the second needs $2$ weights, and the third needs $1$ weight. You can complete the exercise in $4$ operations as follows:

1. Add a weight onto the stack. You do the first exercise.
2. Add a weight onto the stack. You do the second exercise.
3. Remove a weight from the top of the stack. You do the third exercise.
4. Remove a weight from the top of the stack. Now the stack becomes empty.

In Sample Case #2, one way to complete the exercises in $12$ operations is as follows:

1. Add a weight of type $2$.
2. Add a weight of type $3$.
3. Add a weight of type $1$.
4. Add a weight of type $2$. Now the stack contains weights of types $2, 3, 1, 2$ from bottom to top. You do the first exercise.
5. Remove a weight of type $2$ from the top of the stack.
6. Add a weight of type $3$.
7. Add a weight of type $1$. Now the stack contains weights of types $2, 3, 1, 3, 1$ from bottom to top. You do the second exercise.
8. Remove a weight of type $1$ from the top of the stack.
9. Remove a weight of type $3$ from the top of the stack.
10. Remove a weight of type $1$ from the top of the stack.
11. Remove a weight of type $3$ from the top of the stack.
12. Remove a weight of type $2$ from the top of the stack. Now the stack becomes empty.