Bob "the builder" builds integers! He has his own IBM (Integer building machine) to do that. It works like this:

Bob takes an Integer X. For example X is 13. We know that every positive integer can be written as sum of some unique power of 2. So Bob can write 13 as $1 + 4 + 8$. Now 13 can have one of these 3 children:

13+1=14
13+4=17
13+8=21

Similarly children of 12 are:

12+4=16
12+8=20

That's because 12 can be written as 4+8.

Now Bob uses IBM to build children from an Integer. Bob takes X and using the IBM just once he can do following things:

1. Build just one child from X, let's say the child is Y.
2. Build just one child from Y, let's say the child is Z.
3. Build just one child from Z, let's say the child is Q.

And so on...

Ok this can go forever, so Bob never builds an integer larger than a limit L. At each step, Bob can select which child to build. So if the limit is $L = 24$ and $X = 12$, Bob can build 20 from 12 and then 24 from 20 just using the IBM once. Also he can build 16 from 12 using the IBM another time. **One integer can be used multiple times to create new child, but Bob never builds same integer more than once.**

Bob has a list of N integers in his factory. In that list no integer is direct child or even descendant of another. Bob wants to build all possible children from those integers. But using IBM is very costly, it consumes lots of electricity. So your task is to calculate how many times he must use IBM to do it!

Input

Input starts with an integer, T ($T \leq 100$) denoting the number of test cases. Each case has two lines. First line consist two integers N ($1 \leq N \leq 36$) and L ($1 \leq L \leq 10000$). In next line there are N space separated positive integers ($\leq L$) representing the list Bob has in his factory.

Output

For each case, print the case number and the answer, number of times Bob has to use the IBM. See sample output for exact format.

Sample Input

```
2
1 36
20
2 40
8 20
```

Sample Output

```
Case 1: 2
Case 2: 3
```