You are soon to graduate from the mathemagician school of Hagworts, and you're quite content with yourself; all the hard work and elbowing has finally paid off. Being successful at most of your endeavors you would normally end up a herald of sound reasoning and good mathemagics. You, however, are different; not only have you spent your young years secretly hacking away at computers, writing code to do your assigned routine homework for you, but of late you have started planning how to cram all your mathemagical skills into a computer to completely eliminate the need for mathemagicians! To show the others just how great a visionary you are, you plan to make your graduation ceremony something they will never forget.

To do this you need to break into the safe of your arch-nemesis, Hairy Peter. The safe is locked by a code mechanism: All natural numbers from 1 to N need to be typed in in the correct order, set by Hairy Peter. Fortunately you know that Hairy, being a good mathemagician, has a certain weakness; he has a rather unhealthy obsession with the number K. (For instance he always has to introduce himself K times whenever he meets new people, making him quite annoying to be around.) Thus you are certain that his code, when viewed as a permutation of the N first naturals, has order exactly K. (i.e. K is the smallest positive number such that if you K times replace $x \in \{1, ..., N\}$ with the position of x in Hairy's code, you end up with the x you started with, for all x. Thus e.g. the order of the permutation corresponding to the code 2 3 1 is 3, as $1 \to 3 \to 2 \to 1$ and $2 \to 1 \to 3 \to 2$ and $3 \to 2 \to 1 \to 3$.) While this does not help you directly, it greatly reduces the number of code inputs you may have to try before you find the correct one. "How many?" is the question you are pondering right now. You must know the exact number, lest you risk preparing too little time for cracking the safe.

Now you also have a certain mathemagical weakness — arguably a bit worse than Hairy Peter's: Because of your dark scheme to program mathemagical computers, you insist there are no numbers greater than what can be represented by a signed 32-bit integer, namely the prime $P = 2^{31} - 1$. Of course there must be nothing your computers cannot count. In fact you hate this upper limit P so intensely that you have decided to make a new mathemagics where P equals 0. Ha, take that! (Well, you are quite aware that you are really just counting *modulo* P, but it will have to suffice until you find better ways of punishing P.) In fact this turns out to be quite an advantage for you! For instance, if the number of code permutations you have to check turns out to be 2^{31}, there will actually be just one permutation for you to check, as $2^{31} \bmod P = 1$. (Or at least you think so...) That's just magnificent!

Input

The first line of the input file contains an integer T ($T < 30$) which denotes the total number of test cases. The description of each test case is given below:

Input for each set consists of two integers N ($1 \leq N \leq 100$) and K ($1 \leq K \leq 2^{31} - 1$) respectively.

Output

For each case of input, print in a single line the number of permutations of N elements of order K, modulo $2^{31} - 1$. Look at the sample output for details.

Sample Input

```
3
3 2
6 6
15 12
```

Sample Output

```
3
240
1789014075
```