Problem A
Testing 1 2 3
Problem ID: testing
Time Limit: 3 seconds

"Testing, testing... 1, 2, 3. Is this thing on?"

Input

Input consists of a single integer $1 \leq a \leq 9$.

Output

Output a single line containing the text `Testing` followed by a count up to a. See the sample input and output for clarification. There should be exactly one space before each number. Do not print a space after the last number.

Sample Input	Sample Output
5	Testing 1 2 3 4 5

This page is intentionally left blank.

Problem B
Detecting Overflow
Problem ID: overflow
Time Limit: 1 second

Computer hardware represents integers as a fixed length "word". For example, a "signed 32-bit integer" is an integer between -2^{31} and $2^{31}-1$. When using such integers, arithmetic operations may go outside the range of representation.

For example, consider $a = 2000000000$ and $b = 1000000000$. We have $a, b \leq 2^{31}-1$ but $a+b > 2^{31}-1$. If a and b are represented as signed 32-bit integers, the result of the calculation $a+b$ cannot be represented and, actually, the result is probably stored as a negative number! This effect is called `overflow`.

Input

Input consists of just two integers a, b on a single line separated by a single space. You are guaranteed $0 \leq a, b \leq 2^{31}-1$.

Output

Output a single line. If $a+b > 2^{31}-1$, then simply output the text `overflow`. Otherwise, output the value of $a+b$.

Sample Input	Sample Output
`123 456`	`579`

Sample Input	Sample Output
`0 101`	`101`

Sample Input	Sample Output
`2000000000 1000000000`	`overflow`

Sample Input	Sample Output
`1073741824 1073741823`	`2147483647`

Sample Input	Sample Output
`1073741824 1073741824`	`overflow`

This page is intentionally left blank.

Problem C
Factor This!
Problem ID: factor
Time Limit: 3 seconds

People say factoring numbers is hard, and that is why we can trust that certain encryption schemes like RSA and Diffie-Hellman are secure.

I don't know about that. I think factoring is pretty easy. It even made it into the warmup contest!

Input

The first line of input contains a single positive integer $T \leq 100$, denoting the number of test cases. Then, each test case is given on a single line and contains a single integer n. You may be sure that $2 \leq n \leq 100$.

Output

For each test case, output the prime factorization of n on a line. That is, list all primes that divide n and list each one as many times as it divides n. They should be listed in increasing order and consecutive primes should be separated by a space. Do not print a space after the last prime.

Sample Input	Sample Output
3 54 70 7	2 3 3 3 2 5 7 7

This page is intentionally left blank.