

You are required to compute the power of a number by implementing a calculator. Create a class *MyCalculator* which consists of a single method `long power(int, int)`. This method takes two integers, *n* and *p*, as parameters and finds  $n^p$ . If either *n* or *p* is negative, then the method must throw an exception which says "**n or p should not be negative**". Also, if both *n* and *p* are zero, then the method must throw an exception which says "**n and p should not be zero.**"

For example, -4 and -5 would result in

**java.lang.Exception: n or p should not be negative.**

Complete the function `power` in class *MyCalculator* and return the appropriate result after the power operation or an appropriate exception as detailed above.

### Input Format

Each line of the input contains two integers, *n* and *p*. The locked stub code in the editor reads the input and sends the values to the method as parameters.

### Constraints

- $-10 \leq n \leq 10$
- $-10 \leq p \leq 10$

### Output Format

Each line of the output contains the result  $n^p$ , if both *n* and *p* are positive. If either *n* or *p* is negative, the output contains "n and p should be non-negative". If both *n* and *p* are zero, the output contains "n and p should not be zero.". This is printed by the locked stub code in the editor.

### Sample Input 0

```
3 5
2 4
0 0
-1 -2
-1 3
```

### Sample Output 0

```
243
16
java.lang.Exception: n and p should not be zero.
java.lang.Exception: n or p should not be negative.
java.lang.Exception: n or p should not be negative.
```

### Explanation 0

- In the first two cases, both  $n$  and  $p$  are positive. So, the power function returns the answer correctly.
- In the third case, both  $n$  and  $p$  are zero. So, the exception, "n and p should not be zero.", is printed.
- In the last two cases, at least one out of  $n$  and  $p$  is negative. So, the exception, "n or p should not be negative.", is printed for these two cases.