FJCU CPC PUPC 2023 模擬賽 4

12:30~16:30 (SAT) 06/24/2023

ID	Problem Name	Time Limit
A	Vertex	3 secs
B	Prime Factors	3 secs
C	Internet Bandwidth	3 secs
D	Factorial Factors	3 secs
E	The jackpot	3 secs
F	Square Numbers	1 sec
G	One-Two-Three	1 sec
H	Help Dexter	1 sec

Problem A
Vertex

Time limit: 3 seconds

Memory limit: 2048 megabytes

Write a program that searches a directed graph for vertices which are inaccessible from a given starting vertex.

A directed graph is represented by n vertices where $1 \leq n \leq 100$, numbered consecutively $1 \ldots n$, and a series of edges p `->` q which connect the pair of nodes p and q in one direction only.

A vertex r is reachable from a vertex p if there is an edge p `->` r, or if there exists some vertex q for which q is reachable from p and r is reachable from q.

A vertex r is inaccessible from a vertex p if r is not reachable from p.

Input

The input data for this program consists of several directed graphs and starting nodes.

For each graph, there is first one line containing a single integer n . This is the number of vertices in the graph.

Following, there will be a group of lines, each containing a set of integers. The group is terminated by a line which contains only the integer '`0`'. Each set represent a collection of edges. The first integer in the set, i, is the starting vertex, while the next group of integers, $j \ldots k$, define the series of edges i `->` $j \ldots i$ `->` k, and the last integer on the line is always '`0`'. Each possible start vertex i, $1 \leq i \leq n$ will appear once or not at all. Following each graph definition, there will be one line containing a list of integers. The first integer on the line will specify how many integers follow. Each of the following integers represents a start vertex to be investigated by your program. The next graph then follows. If there are no more graphs, the next line of the file will contain only the integer '`0`'.

Output

For each start vertex to be investigated, your program should identify all the vertices which are inaccessible from the given start vertex. Each list should appear on one line, beginning with the count of inaccessible vertices and followed by the inaccessible vertex numbers.

Sample Input

```
3
1 2 0
2 2 0
3 1 2 0
0
2 1 2
0
```

Sample Output

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

Problem B
Prime Factors

Time limit: 3 seconds
Memory limit: 2048 megabytes

An integer $g > 1$ is said to be *prime* if and only if its only positive divisors are itself and one (otherwise it is said to be *composite*). For example, the number 21 is composite; the number 23 is prime. Note that the decompositon of a positive number g into its prime factors, i.e.,

$$g = f_1 \times f_2 \times \cdots \times f_n$$

is unique if we assert that $f_i > 1$ for all i and $f_i \leq f_j$ for $i < j$.

One interesting class of prime numbers are the so-called *Mersenne* primes which are of the form $2^p - 1$. Euler proved that $2^{31} - 1$ is prime in 1772 — all without the aid of a computer.

Input

The input will consist of a sequence of numbers. Each line of input will contain one number g in the range $-2^{31} < g < 2^{31}$, but different of -1 and 1. The end of input will be indicated by an input line having a value of zero.

Output

For each line of input, your program should print a line of output consisting of the input number and its prime factors. For an input number $g > 0$, $g = f_1 \times f_2 \times \cdots \times f_n$, where each f_i is a prime number greater than unity (with $f_i \leq f_j$ for $i < j$), the format of the output line should be

g = f_1 x f_2 x ... x f_n

When $g < 0$, if $| g | = f_1 \times f_2 \times \cdots \times f_n$, the format of the output line should be

g = -1 x f_1 x f_2 x ... x f_n

Sample Input

```
-190
-191
-192
-193
-194
195
196
197
198
199
200
0
```

Sample Output

```
-190 = -1 x 2 x 5 x 19
-191 = -1 x 191
-192 = -1 x 2 x 2 x 2 x 2 x 2 x 2 x 3
-193 = -1 x 193
-194 = -1 x 2 x 97
195 = 3 x 5 x 13
196 = 2 x 2 x 7 x 7
197 = 197
198 = 2 x 3 x 3 x 11
199 = 199
200 = 2 x 2 x 2 x 5 x 5
```

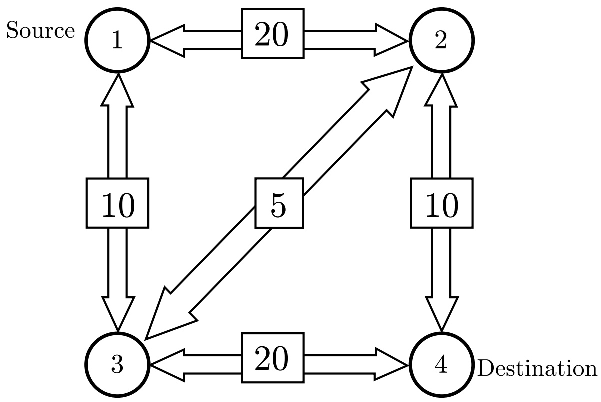
Problem C
Internet Bandwidth

Time limit: 3 seconds

Memory limit: 2048 megabytes

On the Internet, machines (nodes) are richly interconnected, and many paths may exist between a given pair of nodes. The total message-carrying capacity (bandwidth) between two given nodes is the maximal amount of data per unit time that can be transmitted from one node to the other. Using a technique called packet switching, this data can be transmitted along several paths at the same time.

For example, the following figure shows a network with four nodes (shown as circles), with a total of five connections among them. Every connection is labeled with a bandwidth that represents its data-carrying capacity per unit time.



In our example, the bandwidth between node 1 and node 4 is 25, which might be thought of as the sum of the bandwidths 10 along the path 1-2-4, 10 along the path 1-3-4, and 5 along the path 1-2-3-4. No other combination of paths between nodes 1 and 4 provides a larger bandwidth.

You must write a program that computes the bandwidth between two given nodes in a network, given the individual bandwidths of all the connections in the network. In this problem, assume that the bandwidth of a connection is always the same in both directions (which is not necessarily true in the real world).

Input

The input file contains descriptions of several networks. Every description starts with a line containing a single integer n ($2 \le n \le 100$), which is the number of nodes in the network. The nodes are numbered from 1 to n. The next line contains three numbers s, t, and c. The numbers s and t are the source and destination nodes, and the number c is the total number of connections in the network. Following this are c lines describing the connections. Each of these lines contains three integers: the first two are the numbers of the connected nodes, and the third number is the bandwidth of the connection. The bandwidth is a non-negative number not greater than 1000.

There might be more than one connection between a pair of nodes, but a node cannot be connected to itself. All connections are bi-directional, i.e. data can be transmitted in both directions along a connection, but the sum of the amount of data transmitted in both directions must be less than the bandwidth.

A line containing the number '`0`' follows the last network description, and terminates the input.

Output

For each network description, first print the number of the network. Then print the total bandwidth between the source node s and the destination node t, following the format of the sample output. Print a blank line after each test case.

Sample Input

```
4
1 4 5
```

```
1 2 20
1 3 10
2 3 5
2 4 10
3 4 20
0
```

Sample Output

```
Network 1
The bandwidth is 25.
```

Problem D
Factorial Factors

Time limit: 3 seconds

Memory limit: 2048 megabytes

The factorial function, $n! = 1 \cdot 2 \cdot \ldots \cdot n$, has many interesting properties. In this problem, we want to determine the maximum number of integer terms (excluding 1) that can be used to express $n!$. For example:

$$8! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 = 2 \cdot 3 \cdot 2 \cdot 2 \cdot 5 \cdot 3 \cdot 2 \cdot 7 \cdot 2 \cdot 2 \cdot 2 = 2^7 \cdot 3^2 \cdot 5 \cdot 7$$

By inspection, it is clear that the maximum number of terms (excluding 1) that can be multiplied together to produce 8! is 11.

Input

The input for your program consists of a series of test cases on separate lines, ended by end-of-file. Each line contains one number, n, $2 \leq n \leq 1000000$.

Output

For each test case, print the maximum number of factors (excluding 1) that can be multiplied together to produce $n!$. Put the output from each test case on a separate line, starting in the first column.

Sample Input

```
2
1000000
1996
5
8
123456
```

Sample Output

```
1
3626619
5957
5
11
426566
```

Problem E
The jackpot

Time limit: 3 seconds

Memory limit: 2048 megabytes

As Manuel wants to get rich fast and without too much work, he decided to make a career in gambling. Initially, he plans to study the gains and losses of players, so that, he can identify patterns of consecutive wins and elaborate a win-win strategy. But Manuel, as smart as he thinks he is, does not know how to program computers. So he hired you to write programs that will assist him in elaborating his strategy.

Your first task is to write a program that identifies the maximum possible gain out of a sequence of bets. A bet is an amount of money and is either winning (and this is recorded as a positive value), or losing (and this is recorded as a negative value).

Input

The input set consists of a positive number $N \leq 10000$, that gives the length of the sequence, followed by N integers. Each bet is an integer greater than 0 and less than 1000.

The input is terminated with $N = 0$.

Output

For each given input set, the output will echo a line with the corresponding solution. If the sequence shows no possibility to win money, then the output is the message '`Losing streak.`'

Sample Input

```
5
12 -4
-10 4
9
3
-2 -1 -2
0
```

Sample Output

```
The maximum winning streak is 13.
Losing streak.
```

Problem F
Square Numbers

Time limit: 1 second

Memory limit: 2048 megabytes

A square number is an integer number whose square root is also an integer. For example 1, 4, 81 are some square numbers. Given two numbers a and b you will have to find out how many square numbers are there between a and b (inclusive).

Input

The input file contains at most 201 lines of inputs. Each line contains two integers a and b ($0 < a \leq b \leq 100000$). Input is terminated by a line containing two zeroes. This line should not be processed.

Output

For each line of input produce one line of output. This line contains an integer which denotes how many square numbers are there between a and b (inclusive).

Sample Input

```
1 4
1 10
0 0
```

Sample Output

```
2
3
```

Problem G

One-Two-Three

Time limit: 1 second

Memory limit: 2048 megabytes

Your little brother has just learnt to write one, two and three, in English. He has written a lot of those words in a paper, your task is to recognize them. Note that your little brother is only a child, so he may make small mistakes: for each word, there might be at most one wrong letter. The word length is always correct. It is guaranteed that each letter he wrote is in lower-case, and each word he wrote has a unique interpretation.

Input

The first line contains the number of words that your little brother has written. Each of the following lines contains a single word with all letters in lower-case. The words satisfy the constraints above: at most one letter might be wrong, but the word length is always correct. There will be at most 10 words in the input.

Output

For each test case, print the numerical value of the word.

Sample Input

```
3
owe
too
theee
```

Sample Output

```
1
2
3
```

Problem H
Help Dexter

Time limit: 1 second

Memory limit: 2048 megabytes

You know Dexter, right? He is a very talented young scientist. He has a huge lab hidden inside his building. He made all possible security arrangement to keep his naughty sister Dee Dee away from his lab. But she always finds a way into the lab. One day Dee Dee came to the lab and started her usual work, messing up Dexter's lab! Dexter was working on a very important project, so he begged to her and said, "Please!!! Not today. I will do anything for you, but please leave this lab today!!!" Dee Dee was waiting for this chance, she said, "Ok, you do my homework I won't disturb you today." What can Dexter do? He agreed. Dee Dee said, "My teacher told me to write down 17 numbers. First one single digit number, second one two digit number, ..., n-th one n digit number. They will consist of only digit 1 and 2 and the n-th number should be divisible by 2^n." Dexter thought, "I have very little time to finish the project. I can't waste my time for this silly problem, I have bigger problem to think!" So, he sent the modified version of this problem to you. Hurry up, Dee Dee is waiting.

Input

Input starts with an integer T (≤ 300), denoting the number of test cases.

Each case starts with two integers: p q ($1 \leq p, q \leq 17$).

Output

For each case, print the case number first. Then you have to find two integers (smallest and largest) which have p digits and is divisible by 2^q. The integers should contain only `1`'s and `2`'s. If no result is found, print '`impossible`'. If there is only one integer, then print that integer. Otherwise print both integers (first the smallest one then the largest one) separated by a single space.

Sample Input

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

Sample Output

```
Case 1: 12
Case 2: 12 22
Case 3: impossible
```