Al-Khawarizmi 08
Programming Competition

Hosted by International Islamic University
Malaysia

Original Contest was held on 2nd and 3rd July, 2008.
This ranges are slightly different than the original.

You get 13 Pages
10 Problems
&
300 Minutes

A

Square Numbers

Input: Standard Input
Output: Standard Output

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 ($0<a\le b\le 100000$). 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	Output for Sample Input

1 4
1 10
0 0
	2
	3

Problem Setter: Shahriar Manzoor

B

Age Sort

Input: Standard Input
Output: Standard Output

You are given the ages (in years) of all people of a country with at least 1 year of age. You know that no individual in that country lives for 100 or more years. Now, you are given a very simple task of sorting all the ages in ascending order.

Input

There are multiple test cases in the input file. Each case starts with an integer **n** (0<**n**<=2000000), the total number of people. In the next line, there are **n** integers indicating the ages. Input is terminated with a case where **n** = 0. This case should not be processed.

Output

For each case, print a line with **n** space separated integers. These integers are the ages of that country sorted in ascending order.

Warning: Input Data is pretty big (~ 25 MB) so use faster IO.

Sample Input

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

Output for Sample Input

```
1 2 3 4 5
1 2 2 3 3
```

The memory limit for this problem is 2 Megabyte only.

Problem Setter: Mohammad Mahmudur Rahman
Special Thanks: Shahriar Manzoor

C

Commandos

Input: Standard Input
Output: Standard Output

A group of commandos were assigned a critical task. They are to destroy an enemy head quarter. The enemy head quarter consists of several buildings and the buildings are connected by roads. The commandos must visit each building and place a bomb at the base of each building. They start their mission at the base of a particular building and from there they disseminate to reach each building. The commandos must use the available roads to travel between buildings. Any of them can visit one building after another, but they must all gather at a common place when their task in done. In this problem, you will be given the description of different enemy headquarters. Your job is to determine the minimum time needed to complete the mission. Each commando takes exactly one unit of time to move between buildings. You may assume that the time required to place a bomb is negligible. Each commando can carry unlimited number of bombs and there is an unlimited supply of commando troops for the mission.

Input

The first line of input contains a number $\mathbf{T<50}$, where **T** denotes the number of test cases. Each case describes one head quarter scenario. The first line of each case starts with a positive integer $\mathbf{N\le100}$, where **N** denotes the number of buildings in the head quarter. The next line contains a positive integer **R,** where **R** is the number of roads connecting two buildings. Each of the next **R** lines contain two distinct numbers, $\mathbf{0\le u,v<N}$, this means there is a road connecting building **u** to building **v.** The buildings are numbered from **0** to **N-1.** The last line of each case contains two integers $\mathbf{0\le s,d<N}$. Where **s** denotes the building from where the mission starts and **d** denotes the building where they must meet.

You may assume that two buildings will be directly connected by at most one road. The input will be such that, it will be possible to go from any building to another by using one or more roads.

Output

For each case of input, there will be one line of output. It will contain the case number followed by the minimum time required to complete the mission. Look at the sample output for exact formatting.

Sample Input

```
2
4
3
0 1
2 1
1 3
0 3
2
1
0 1
1 0
```

Output for Sample Input

```
Case 1: 4
Case 2: 1
```

Problem Setter: Shamim Hafiz
Special Thanks: Md. Arifuzzaman Arif

D

Even Parity

Input: Standard Input
Output: Standard Output

We have a grid of size N x N. Each cell of the grid initially contains a zero(0) or a one(1).
The *parity* of a cell is the number of 1s surrounding that cell. A cell is surrounded by at most 4 cells (top, bottom, left, right).

Suppose we have a grid of size 4 x 4:

1	0	1	0
1	1	1	1
0	1	0	0
0	0	0	0

The parity of each cell would be

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

For this problem, you have to change some of the 0s to 1s so that the parity of every cell becomes even. We are interested in the minimum number of transformations of 0 to 1 that is needed to achieve the desired requirement.

Input

The first line of input is an integer **T** (**T<30**) that indicates the number of test cases. Each case starts with a positive integer **N**(**1≤N≤15**). Each of the next **N** lines contain **N** integers (**0/1**) each. The integers are separated by a single space character.

Output

For each case, output the case number followed by the minimum number of transformations required. If it's impossible to achieve the desired result, then output -1 instead.

Sample Input

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

Output for Sample Input

```
Case 1: 0
Case 2: 3
Case 3: -1
```

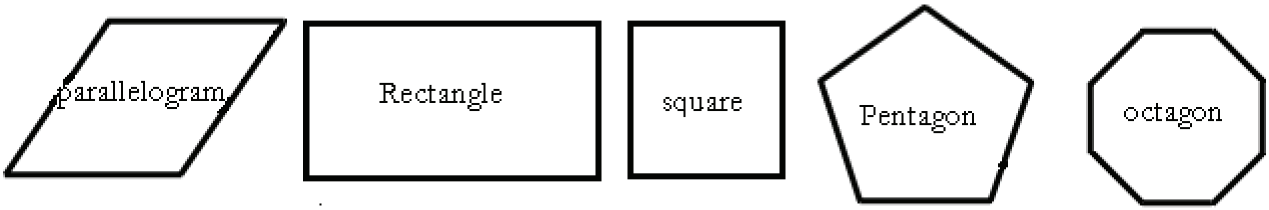
Problem Setter: Sohel Hafiz,
Special Thanks: Derek Kisman, Md. Arifuzzaman Arif

E

Count the Polygons

Input: Standard Input
Output: Standard Output

A polygon is a plane figure that is bounded by a closed path and composed of a finite sequence of straight line segments. These segments are called its edges, and the points where two edges meet are the polygon's vertices.



You have got a set of N sticks of various lengths. How many ways can you choose K sticks from this set and form a polygon with K sides by joining the end points.

Input

The first line of input is an integer **T** (**T<100**) that indicates the number of test cases. Each case starts with a line containing 2 positive integers **N** and **K** (**3≤N≤30 & 3≤K≤N**). The next line contains **N** positive integers in the range [**1, 2^31)**, which represents the lengths of the available sticks. The integers are separated by a single space.

Output

For each case, output the case number followed by the number of valid polygons that can be formed by picking K sticks from the given N sticks.

Sample Input

```
4
4 3
10 10 20 20
6 4
1 1 1 1 1 1
4 3
10 20 30 100000000
6 6
2 3 4 5 6 7
```

Output for Sample Input

```
Case 1: 2
Case 2: 15
Case 3: 0
Case 4: 1
```

Problem Setter: Sohel Hafiz
Special Thanks: Md. Arifuzzaman Arif

F

Largest Prime Divisor

Input: Standard Input
Output: Standard Output

All integer numbers are divisible by primes. If a number is divisible by more than one prime number, then it obviously has a largest prime divisor. The numbers which do not fall in this category do not have a largest prime divisor. Given a number N your job is to write a program that finds its largest prime divisor. An integer number n is divisible by another integer number m if there is an integer t such that mt=n.

Input

The input file contains at most 450 sets of inputs. Each line contains a decimal integer N. N does not have more than 14 digits. Input is terminated by a line containing a single zero. So no other line except the last line contains a zero in the input. This line need not be processed.

Output

For each line of the input produce one line of output. This line contains an integer LPD, which is the largest prime divisor of the input number N. If the input number is not divisible by more than one prime number output a -1.

Sample Input

```
2
6
100
0
```

Output for Sample Input

```
-1
3
5
```

Problem Setter: Shahriar Manzoor
Special Thanks: Syed Monowar Hossain

G

Pythagorean Triangles

Input: Standard Input
Output: Standard Output

Many lattice triangles are formed in an (NxN) grid, but not all of them are Pythagorean (Right angled) triangles. Given the value of N your job is to write a program that produces the number of lattice triangles in an (NxN) grid. A lattice triangle is triangle whose three vertices are lattice points. A lattice point in two dimensional Cartesian coordinate system is a point whose abscissa and ordinate are integers.

Input

The input file contains at most 15 lines of inputs. Each line contains an integer N (0<N<2001). Input is terminated by a line containing a single zero. This line should not be processed.

Output

For each value of N produce one line of output which contains an integer T. Here T denotes the total number of right angled triangles in that (NxN) grid.

Sample Input

```
10
20
30
0
```

Output for Sample Input

```
23596
418716
2288304
```

Problem Setter: Shahriar Manzoor
Special Thanks: Derek Kisman

H

Substring

Input: Standard Input
Output: Standard Output

Given a set of pattern strings, and a text, you have to find, if any of the pattern is a substring of the text. If any of the pattern string can be found in text, then print “yes”, otherwise “no” (without quotes). But, unfortunately, that’s not what is asked here. ☺
The problem described above, requires a input file generator. The generator generates a text of length L, by choosing L characters randomly. Probability of choosing each character is given as priori, and independent of choosing others.
Now, given a set of patterns, calculate the probability of a valid program generating “no”.

Input

First line contains an integer T, the number of test cases. Each case starts with an integer K, the number of pattern strings. Next K lines each contain a pattern string, followed by an integer N, number of valid characters. Next N lines each contain a character and the probability of selecting that character, p_i. Next an integer L, the length of the string generated. The generated text can consist of only the valid characters, given above.

There will be a blank line after each test case.

Output

For each test case, output the number of test case, and the probability of getting a “no”.

Constraints

- $T \le 50$
- $K \le 20$
- Length of each pattern string is between 1 and 20
- Each pattern string consists of only alphanumeric characters (‘a’ to ‘z’, ‘A’ to ‘Z’,’0’ to ‘9’)
- Valid characters are all alphanumeric characters
- $\sum p_i = 1$
- $L \le 100$

Sample Input

```
2
1
a
2
a 0.5
b 0.5
2

2
ab
```

Output for Sample Input

```
Case #1: 0.250000
Case #2: 0.840000
```

`ab`
`2`
`a 0.2`
`b 0.8`
`2`	

Problem Setter: Manzurur Rahman Khan

I

Treasure Hunt

Input: Standard Input
Output: Standard Output

Famous Archaeologist Dr. Asiana Jones has discovered something extraordinary. He found a thousand years old tomb, and discovered that there are thousands of treasure rooms beneath it. Each treasure room contains a chest full of treasure. There can be a number of tunnels to go from one room to another. But, since the tunnels are thousand years old(like the tomb itself), they are very fragile. You have to be very quick, and you can't use a tunnel more than once.

Dr. Asiana Jones has managed to calculate the exact positions of all the tombs. But, as the rooms are beneath the tomb, he must dig a hole, one the floor to get to any of them. He doesn't want to dig more than one hole, because, otherwise, the whole tomb may become unstable.

If Dr. Jones can start from any of the treasure room, find the maximum number of treasure he can collect. Dr. Jones has to start and end at the same room, since, that's the only way to get out of the tomb.

Input

First line of input contains an integer T, the number of test cases, followed by T scenarios. Each scenario starts with two integers, N and M, the number of treasure rooms, and the number of tunnels. Each treasure room is uniquely identified by an integer between 1 and N. Each of the following M lines each, contain two integers, the treasure rooms at the end of the corresponding tunnel. The tunnels can be traversed in any direction, but at most once. That is, if there is a tunnel between room 1 and 2, it can be used to move from room 1 to 2, or 2 to 1. But, after any of these move, you can't use this tunnel any more. There will be a blank line before each scenario.

Output

For each scenario, print the scenario number, followed by the maximum number of treasure to collect. Next line will contain a sequence of integers, the starting rooms, from where, you can collect maximum number of treasures.

Constraints

- $T \leq 31$
- $1 \leq N \leq 10{,}000$
- $0 \leq M \leq 1{,}000{,}000$

Sample Input

```
3

4 3
1 2
2 3
1 4

3 3
1 2
```

Output for Sample Input

```
Case #1: 1
1 2 3 4
Case #2: 3
1 2 3
Case #3: 3
2 3 4
```

`2 3`
`3 1`
 	
`4 4`
`1 2`
`2 3`
`3 4`
`4 2`	

Problem Setter: Manzurur Rahman Khan
Special Thanks: Abdullah-al-Mahmud

J

Square Sums

Input: Standard Input
Output: Standard Output

Do you know that there are squares within a square. This might seem confusing, but take a look at this.

Suppose you have a square grid of size 5 X 5 completely filled with integers.

5	**3**	**2**	**7**	**9**
1	*7*	*4*	*2*	**4**
5	*3*	**2**	*4*	**6**
1	*3*	*4*	*5*	**1**
1	**4**	**5**	**6**	**3**

You can make three squares from the table above… well there are more but we will consider only concentric squares. The squares are denoted using different fonts.
In this problem you have to find the sum of the numbers of each square.
For the above case the sums are
5 + 3 + 2 + 7 + 9 + 1 + 4 + 5 + 6 + 1 + 1 + 1 + 4 + 5 + 6 + 3 = 63
7 + 4 + 2 + 3 + 4 + 3 + 4 + 5 = 32
2 = 2

Input

There will be several lines in the input file. Each case starts with an integer **n(n<=10)** that determines the dimension of the square grid. Each of the next **n** lines will contain **n** integers each that will fill the square table in row major order. Input is terminated by a case where **n** == 0.

Output

Each line of output will start with "`Case #:`" where # is replaced by the case number. Then you have to output **ceil(n/2)** space separated numbers that will represent the sums from outer to inner. Follow the sample for exact details.

Sample Input

```
5
5 3 2 7 9
1 7 4 2 4
5 3 2 4 6
1 3 4 5 1
1 4 5 6 3
1
1
0
```

Output for Sample Input

```
Case 1: 63 32 2
Case 2: 1
```

Problem Setter: Sohel Hafiz