Continuous Assessment: 02

CSE,RU

Answer any TWO

1.	N points of a polygon are given. How can you efficiently find out, if a polygon is convex or non-convex?	10
2.	Given a set of numbers. Perform greatest common divisor (GCD) operation to any range of the numbers. Numbers are {2, 3, 60, 90, 50} and queries are given by index ranges {(1,3), (2,4) and (0,2)}. Represent the numbers using an efficient data structure and perform the queries.	10
3.	How can you find the largest circle that can fit inside a non-convex polygon?	10

Continuous Assessment: 01

CSE,RU

Answer any TWO

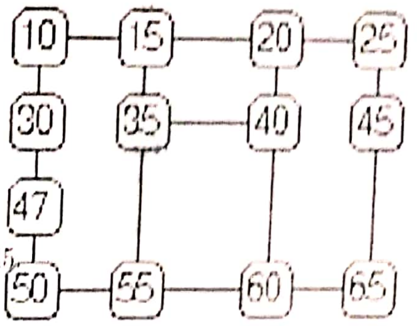
1. You have to set a problem for a competitive programming contest. Write a problem description so that it can be solved using Depth-First-Search algorithm. Prepare sample input-output and judge input-output. Judge input-output should contain at least one critical input-output. Which one is critical input-output and why, Explain. 10
2. Given some numbers 31, 20, 30, 25, 14, 8, 21, 9, 26, 12, 28, 14, 7, 27, 3, 11, 19 and 2. Develop an algorithm so that it can sort the numbers in O(n) time. You can't use any array. 10
3. In terms of time complexity and space complexity, which input representation is suitable for Breath-First-Search algorithm and why,,Explain. 10

To avoid the potential problem of network messages (packets) looping around forever inside a network, each message includes a Time To Live (TTL) field. This field contains the number of nodes (stations, computers, etc.) that can retransmit the message, forwarding it along toward its destination, before the message is unceremoniously dropped. Each time a station receives a message it decrements the TTL field by 1. If the destination of the message is the current station, then the TTL field's value is ignored. However, if the message must be forwarded, and the decremented TTL field contains zero, then the message is not forwarded.

In this problem you are given the description of a number of networks, and for each network you are asked to determine the number of nodes that are not reachable given an initial node and TTL field value.

Consider the example network on the right:

If a message with a TTL field of 2 was sent from node 35 it could reach nodes 15, 10, 55, 50, 40, 20 and 60. It could not reach nodes 30, 47, 25, 45 or 65, since the TTL field would have been set to zero on arrival of the message at nodes 10, 20, 50 and 60. If we increase the TTL field's initial value to 3, starting from node 35 a message could reach all except node 45.



Input

There will be multiple network configurations provided in the input. Each network description starts with an integer *NC* specifying the number of connections between network nodes. An *NC* value of zero marks the end of the input data. Following *NC* there will be *NC* pairs of positive integers. These pairs identify the nodes that are connected by a communication line. There will be no more than one (direct) communication line between any pair of nodes, and no network will contain more than 30 nodes. Following each network configuration there will be multiple queries as to how many nodes are not reachable given an initial node and TTL field setting. These queries are given as a pair of integers, the first identifying the starting node and the second giving the initial TTL field setting. The queries are terminated by a pair of zeroes.

Output

For each query display a single line showing the test case number (numbered sequentially from one), the number of nodes not reachable, the starting node number, and the initial TTL field setting. The sample input and output shown below illustrate the input and output format.

Sample Input

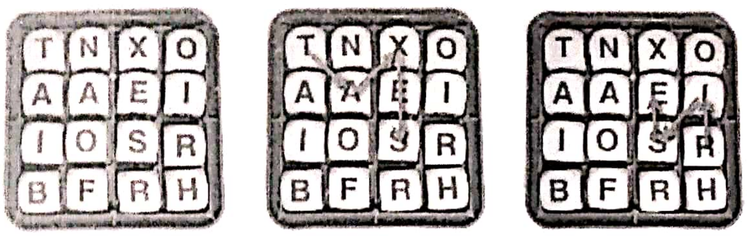
```
16
10 15    15 20    20 25    10 30    30 47    47 50    25 45    45 65
15 35    35 55    20 40    50 55    35 40    55 60    40 60    60 65
35  2    35  3    ~~0 0~~
                55 2   0 0

14
1 2   2 7   1 3   3 4   3 5   5 10   5 11
4 6   7 6   7 8   7 9   8 9   8  6   6 11
1 1   1 2   3 2   3 3   0 0

0
```

Sample Output

```
Case 1: 5 nodes not reachable from node 35 with TTL = 2.
Case 2: 1 nodes not reachable from node 35 with TTL = 3.
Case 3: 8 nodes not reachable from node 1 with TTL = 1.
Case 4: 5 nodes not reachable from node 1 with TTL = 2.
Case 5: 3 nodes not reachable from node 3 with TTL = 2.
Case 6: 1 nodes not reachable from node 3 with TTL = 3.
```

Boggle® is a classic word game played on a 4 by 4 grid of letters. The letter grid is randomly generated by shaking 16 cubes labeled with a distribution of letters similar to that found in English words. Players try to find words hidden within the grid.

Words are formed from letters that adjoin horizontally, vertically, or diagonally. However, no letter may be used more than once within a single word.



An example Boggle® letter grid, showing the formation of the words "taxes" and "rise".

The score awarded for a word depends on its length, with longer words being worth more points. Exact point values are shown in the table below. A word is only ever scored once, even if it appears multiple times in the grid.

No. of letters:	3	4	5	6	7	8 or more
Points:	1	1	2	3	5	11

In this problem, your task is to write a program that plays Boggle® . Given a letter grid and a dictionary of words, you are to calculate the total score of all the words in the dictionary that can be found in the grid.

Input

The first line of the input file contains a number N, the number of Boggle® games that follow.

Each Boggle game begins with 16 capital letters arranged in a 4 by 4 grid, representing the board configuration for that game. A blank line always precedes the letter grid. Following the letter grid is a single number M ($1 \leq M \leq 100$), the number of words in your dictionary for that game. The next M lines contain the dictionary words, one per line, in no particular order. Each word consists of between 3 and 16 capital letters. No single word will appear in the dictionary more than once for a given Boggle® game.

Output

For each Boggle® game in the input, your program should output the total score for that game. Follow the format given in the sample output.

Sample Input

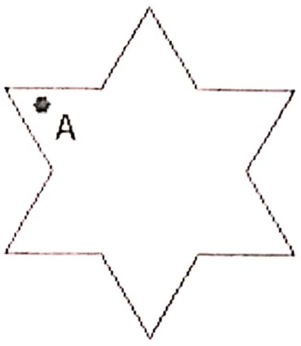
```
2

TNXO
AAEI
IOSR
BFRH
8
TAXES
RISE
ANNEX
BOAT
OATS
FROSH
HAT
TRASH

FNEI
OBCN
EERI
VSIR
1
BEER
```

Sample Output

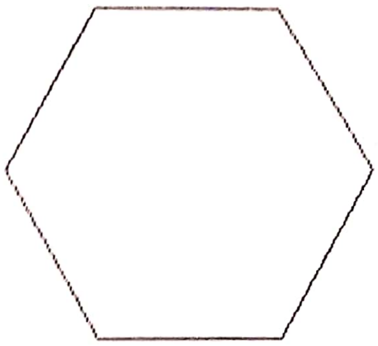
```
Score for Boggle game #1: 6
Score for Boggle game #2: 1
```

Century Arts has hundreds of art galleries scattered all around the country and you are hired to write a program that determines whether any of the galleries has a *critical point*. The galleries are polygonal in shape and a critical point is a point inside that polygon from where the entire gallery is not visible.

For example, in gallery 1 (drawn below) point A is a critical point, but gallery 2 has no critical point at all.



Gallery 1



Gallery 2

The *Century Arts* authorities will provide you with the shape of a gallery as a sequence of (x, y) co-ordinates (determined using a suitable origin) of the consecutive corner points of that gallery.

Input

The input file consists of several data blocks. Each data block describes one gallery.

The first line of a data block contains an integer N ($3 \le N \le 50$) indicating the number of corner points of the gallery. Each of the next N lines contains two integers giving the (x, y) co-ordinates of a corner point where $0 \le x, y \le 1000$. Starting from the first point given in the input the corner points occur in the same order on the boundary of the gallery as they appear in the input. No three consecutive points are co-linear.

The input file terminates with a value of `0` for N.

Output

For each gallery in the input output the word '`Yes`' if the gallery contains a critical point, otherwise output the word '`No`'. Each output must be on a separate line.

Sample Input

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

9
0 0
0 9
5 5
9 0

Sample Output

```
No
Yes
```

A set of laboratory mice is being trained to escape a maze. The maze is made up of cells, and each cell is connected to some other cells. However, there are obstacles in the passage between cells and therefore there is a time penalty to overcome the passage Also, some passages allow mice to go one-way, but not the other way round.

Suppose that all mice are now trained and, when placed in an arbitrary cell in the maze, take a path that leads them to the exit cell in minimum time.

We are going to conduct the following experiment: a mouse is placed in each cell of the maze and a count-down timer is started. When the timer stops we count the number of mice out of the maze.

Write a program that, given a description of the maze and the time limit, predicts the number of mice that will exit the maze. Assume that there are no bottlenecks is the maze, i.e. that all cells have room for an arbitrary number of mice.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The maze cells are numbered $1, 2, \ldots, N$, where N is the total number of cells. You can assume that $N \leq 100$.

The first three input lines contain N, the number of cells in the maze, E, the number of the exit cell, and the starting value T for the count-down timer (in some arbitrary time unit).

The fourth line contains the number M of connections in the maze, and is followed by M lines, each specifying a connection with three integer numbers: two cell numbers a and b (in the range $1, \ldots, N$) and the number of time units it takes to travel from a to b.

Notice that each connection is one-way, i.e., the mice can't travel from b to a unless there is another line specifying that passage. Notice also that the time required to travel in each direction might be different.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output consists of a single line with the number of mice that reached the exit cell E in at most T time units.

Sample Input

```
1

4
2
1  2
8
1 2 1
1 3 1
2 1 1
2 4 1
3 1 1
3 4 1
4 2 1 2
4 3 1
```

3
9

Sample Output

```
3
```

9