A • Nth Largest Value

For this problem, you will write a program that prints the Nth largest value in a fixed sized array of integers. To make things simple, **N** will be 3 and the array will always be have 10 decimal integer values.

Input

The first line of input contains a single integer **P**, ($1 \le P \le 1000$), which is the number of data sets that follow. Each data set consists of a single line containing the data set number, followed by a space, followed by 10 space separated decimal integers whose values are between 1 and 1000 inclusive.

Output

For each data set, generate one line of output with the following values: The data set number as a decimal integer, a space, and the 3rd largest value of the corresponding 10 integers.

Sample Input	Sample Output
4	1 8
1 1 2 3 4 5 6 7 8 9 1000	2 489
2 338 304 619 95 343 496 489 116 98 127	3 931
3 931 240 986 894 826 640 965 833 136 138	4 768
4 940 955 364 188 133 254 501 122 768 408	

acm 2009

Greater New York
Programming Contest

Hofstra University
Hempstead, NY

event sponsor

IBM

B • Equal Sum Partitions

An *equal sum partition* of a sequence of numbers is a grouping of the numbers (in the same order as the original sequence) in such a way that each group has the same sum. For example, the sequence:

2 5 1 3 3 7

may be grouped as:

(2 5) (1 3 3) (7)

to yield an equal sum of **7**.

Note: The partition that puts all the numbers in a single group is an equal sum partition with the sum equal to the sum of all the numbers in the sequence.

For this problem, you will write a program that takes as input a sequence of positive integers and returns the *smallest* sum for an equal sum partition of the sequence.

Input

The first line of input contains a single integer P, $(1 \leq P \leq 1000)$, which is the number of data sets that follow. The first line of each data set contains the data set number, followed by a space, followed by a decimal integer M, $(1 \leq M \leq 10000)$, giving the total number of integers in the sequence. The remaining line(s) in the dataset consist of the values, 10 per line, separated by a single space. The last line in the dataset may contain less than 10 values.

Output

For each data set, generate one line of output with the following values: The data set number as a decimal integer, a space, and the smallest sum for an equal sum partition of the sequence.

Sample Input	Sample Output
3	1 7
1 6	2 21
2 5 1 3 3 7	3 2
2 6	
1 2 3 4 5 6	
3 20	
1 1 2 1 1 2 1 1 2 1	
1 2 1 1 2 1 1 2 1 1	

C • Balls

The classic *Two Glass Balls* brain-teaser is often posed as:

> "Given two identical glass spheres, you would like to determine the lowest floor in a
> 100-story building from which they will break when dropped. Assume the spheres are
> undamaged when dropped below this point. What is the strategy that will minimize the
> worst-case scenario for number of drops?"

Suppose that we had only one ball. We'd have to drop from each floor from 1 to 100 in sequence,
requiring 100 drops in the worst case.

Now consider the case where we have two balls. Suppose we drop the first ball from floor n. If it
breaks we're in the case where we have one ball remaining and we need to drop from floors 1 to n-1
in sequence, yielding n drops in the worst case (the first ball is dropped once, the second at most n-1
times). However, if it does not break when dropped from floor n, we have reduced the problem to
dropping from floors n+1 to 100. In either case we must keep in mind that we've already used one
drop. So the minimum number of drops, in the worst case, is the minimum over all n.

You will write a program to determine the minimum number of drops required, in the worst case,
given B balls and an M-story building.

Input

The first line of input contains a single integer P, $(1 \le P \le 1000)$, which is the number of data sets that
follow. Each data set consists of a single line containing three (3) decimal integer values: the
problem number, followed by a space, followed by the number of balls B, $(1 \le B \le 50)$, followed by a
space and the number of floors in the building M, $(1 \le M \le 1000)$.

Output

For each data set, generate one line of output with the following values: The data set number as a
decimal integer, a space, and the minimum number of drops needed for the corresponding values of
B and M.

Sample Input	Sample Output
4	1 4
1 2 10	2 14
2 2 100	3 24
3 2 300	4 10
4 25 900	

acm 2009

Greater New York
Programming Contest
Hofstra University
Hempstead, NY

event sponsor

IBM.

D • Running Median

For this problem, you will write a program that reads in a sequence of 32-bit *signed* integers. After each *odd-indexed* value is read, output the *median* (middle value) of the elements received so far.

Input

The first line of input contains a single integer *P*, (1 ≤ *P* ≤ 1000), which is the number of data sets that follow. The first line of each data set contains the data set number, followed by a space, followed by an odd decimal integer *M*, (1 ≤ *M* ≤ 9999), giving the total number of signed integers to be processed. The remaining line(s) in the dataset consists of the values, 10 per line, separated by a single space. The last line in the dataset may contain less than 10 values.

Output

For each data set the first line of output contains the data set number, a single space and the number of medians output (which *should* be one-half the number of input values plus one). The output medians will be on the following lines, 10 per line separated by a single space. The last line may have less than 10 elements, but at least 1 element. There should be no blank lines in the output.

Sample Input	Sample Output
3	1 5
1 9	1 2 3 4 5
1 2 3 4 5 6 7 8 9	2 5
2 9	9 8 7 6 5
9 8 7 6 5 4 3 2 1	3 12
3 23	23 23 22 22 13 3 5 5 3 -3
23 41 13 22 -3 24 -31 -11 -8 -7	-7 -3
3 5 103 211 -311 -45 -67 -73 -81 -99	
-33 24 56	

E • The Next Permutation

For this problem, you will write a program that takes a (possibly long) string of decimal digits, and outputs the permutation of those decimal digits that has the next *larger* value (as a decimal number) than the input number. For example:

```
123 -> 132
279134399742 -> 279134423799
```

It is possible that no permutation of the input digits has a larger value. For example, **987**.

Input

The first line of input contains a single integer **P**, (1 ≤ **P** ≤ 1000), which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by up to 80 decimal digits which is the input value.

Output

For each data set there is one line of output. If there is no larger permutation of the input digits, the output should be the data set number followed by a single space, followed by the string **BIGGEST**. If there is a solution, the output should be the data set number, a single space and the next larger permutation of the input digits.

Sample Input	Sample Output
3	1 132
1 123	2 279134423799
2 279134399742	3 BIGGEST
3 987	

F • Adjacent Bit Counts

For a string of **n** bits $x_1, x_2, x_3, ..., x_n$, the *adjacent bit count* of the string (`AdjBC(x)`) is given by

$$x_1{*}x_2 \;+\; x_2{*}x_3 \;+\; x_3{*}x_4 \;+\; ... \;+\; x_{n-1}{*}x_n$$

which counts the number of times a 1 bit is adjacent to another 1 bit. For example:

```
AdjBC(011101101) = 3
AdjBC(111101101) = 4
AdjBC(010101010) = 0
```

Write a program which takes as input integers **n** and **k** and returns the number of bit strings **x** of **n** bits (out of 2^n) that satisfy `AdjBC(x) = k`. For example, for 5 bit strings, there are 6 ways of getting `AdjBC(x) = 2`:

```
11100, 01110, 00111, 10111, 11101, 11011
```

Input

The first line of input contains a single integer **P**, $(1 \le P \le 1000)$, which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by a decimal integer giving the number (**n**) of bits in the bit strings, followed by a single space, followed by a decimal integer (**k**) giving the desired adjacent bit count. The number of bits (**n**) will not be greater than 100 and the parameters **n** and **k** will be chosen so that the result will fit in a *signed* 32-bit integer.

Output

For each data set there is one line of output. It contains the data set number followed by a single space, followed by the number of **n**-bit strings with adjacent bit count equal to **k**.

Sample Input	Sample Output
10	1 6
1 5 2	2 63426
2 20 8	3 1861225
3 30 17	4 168212501
4 40 24	5 44874764
5 50 37	6 160916
6 60 52	7 22937308
7 70 59	8 99167
8 80 73	9 15476
9 90 84	10 23076518
10 100 90	

acm 2009
Greater New York Programming Contest
Hofstra University
Hempstead, NY
event sponsor
IBM

G • Convex Hull of Lattice Points

A **lattice point** is a point with **integer** coordinates. A **lattice polygon** is a polygon with all vertices lattice points.

A polygon is **convex** if any line segment between two points of the polygon is inside (or on the boundary of) the polygon. Equivalently, the interior angle at each polygon vertex is less than 180 degrees.

For a set **S**, of lattice points, the **convex hull** is the smallest convex (lattice) polygon which contains all points of the set. (The vertices of the convex hull must be members of the set of lattice points). If all the points are on a single straight line, the convex hull will be a line segment (a **degenerate** polygon – see rightmost diagram below). In the diagrams below, the points of the set are indicated by solid dots, the vertices of the convex hull by **X**'s and the convex hull is drawn connecting the vertices. Note that not all points on the convex hull polygon are vertices.
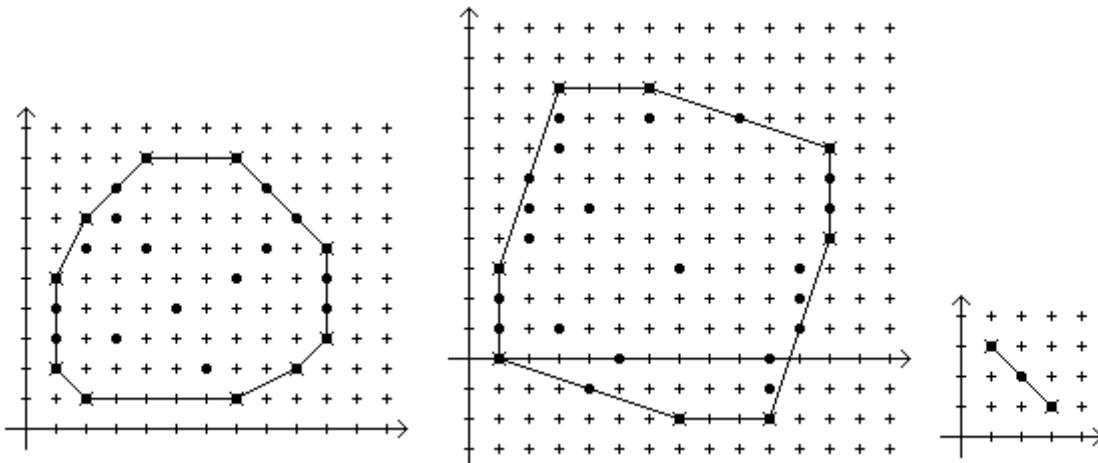
The vertices of a lattice polygon are in *standard order* if:
  a) The first vertex is the one with the largest **y** value. If two vertices have the same **y** value, the one with the smaller **x** value is the first.
  b) Vertices are given in clockwise order around the polygon.

Write a program that reads a set of lattice points and outputs the vertices of the convex hull of the points in *standard order*.

acm 2009

Greater New York
Programming Contest
Hofstra University
Hempstead, NY

event sponsor
IBM.

Input

The first line of input contains a single integer *P*, (1 ≤ *P* ≤ 1000), which is the number of data sets that follow. The first line of each data set contains the data set number, followed by a space, followed by a decimal integer giving the number of points *N*, (3 ≤ *N* ≤ 50), in the set. The remaining lines in the data set contain the points in the set, at most 5 points per line (the last line may have fewer). Each point consists of 2 space separated decimal integer values representing the *x* and *y* coordinates respectively.

Output

For each data set there are multiple lines of output. The first line contains a decimal integer giving the data set number followed by a single space, followed by a decimal integer giving the total number of vertices of the convex hull. The vertices of the convex hull follow, one per line in standard order. Each line contains the decimal integer *x* coordinate, a single space and the decimal integer *y* coordinate.

Sample Input	Sample Output
4	1 10
1 25	4 9
2 1 7 1 1 2 9 2 1 3	7 9
10 3 1 4 10 4 1 5 10 5	10 6
2 6 10 6 2 7 9 7 3 8	10 3
8 8 4 9 7 9 6 2 3 3	9 2
5 4 7 5 8 6 4 6 3 7	7 1
2 30	2 1
3 9 6 9 3 8 9 8 3 7	1 2
12 7 2 6 12 6 2 5 12 5	1 5
2 4 12 4 1 3 11 3 1 2	2 7
11 2 1 1 11 1 1 0 10 0	2 8
4 -1 10 -1 7 -2 10 -2 5 0	3 9
7 3 4 5 6 8 3 1 2 6	6 9
3 3	12 7
3 1 2 2 1 3	12 4
4 6	10 -2
1 3 19 1 4 2 2 1 11 2	7 -2
10 1	1 0
	1 3
	3 2
	1 3
	3 1
	4 4
	1 3
	11 2
	19 1
	2 1

# H ● Interior Points of Lattice Polygons

A **lattice point** is a point with **integer** coordinates. A **lattice polygon** is a polygon with all vertices lattice points.
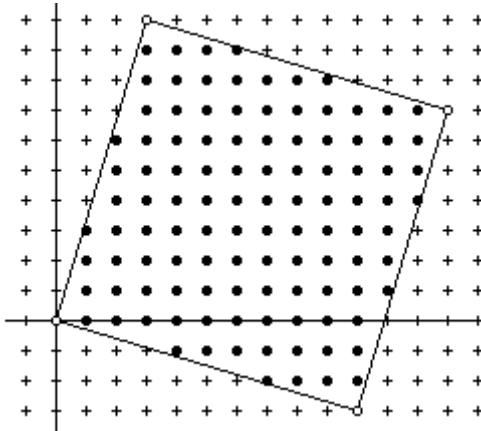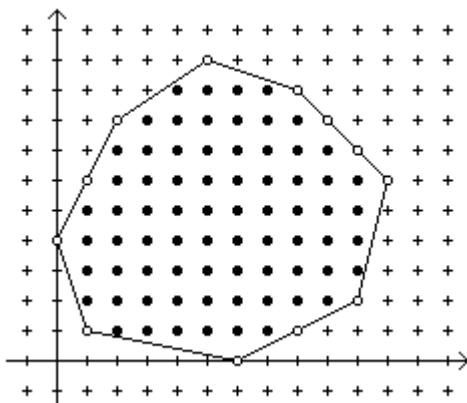
The lattice points on the boundary of the polygon are **boundary points** (open dots in the figure above) and the points inside and not on the polygon are **interior points** (filled in dots in the figure above).

A polygon is **convex** if any line segment between two points of the polygon is inside (or on the boundary of) the polygon. Equivalently, the interior angle at each polygon vertex is less than 180 degrees. Note that any line between two points inside (and not on the boundary of) the polygon is entirely inside (and not on the boundary of) the polygon.

The interior points of a convex lattice polygon on any horizontal line form a single segment from a leftmost point to a rightmost point (which may be the same). Note that there may be no interior points (A), or only one (B), or isolated points (C) as shown in the figures below.

A

B

C
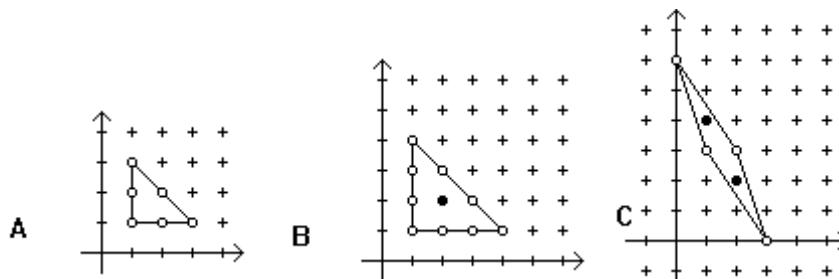
Write a program that reads the vertices of a convex lattice polygon in standard order and outputs the interior points as a list of horizontal line segments. The vertices of a lattice polygon are in *standard order* if:

    a) The first vertex is the one with the largest *y* value. If two vertices have the same *y* value, the one with the smaller *x* value is the first.

    b) Vertices are given in clockwise order around the polygon.

acm 2009

Greater New York
Programming Contest

Hofstra University
Hempstead, NY

event sponsor

IBM.

Input

The first line of input contains a single integer **P**, (1 ≤ **P** ≤ 1000), which is the number of data sets that follow. The first line of each data set contains the data set number, followed by a space, followed by a decimal integer giving the number vertices **N**, (3 ≤ **N** ≤ 50), of the polygon. The remaining lines in the data set contain the vertices, one per line in standard order. Each line contains the decimal integer **x** coordinate, a space and the decimal integer **y** coordinate.

Output

For each data set there are multiple lines of output. The first line contains a decimal integer giving the data set number followed by a single space, followed by a decimal integer giving the number of horizontal lines which contain interior points (this may be zero (0) or more). The lines of interior points, if any, follow, one per line in order of *decreasing y* value. Each line contains the decimal integer **y** coordinate, a single space and the decimal integer **x** coordinate of the left most point, a single space and the decimal integer **x** coordinate of the right most point.

| Sample Input	Sample Output
6	1 9
1 8	9 4 7
5 10	8 3 8
8 9	7 2 9
11 6	6 2 10
10 2	5 1 10
6 0	4 1 10
1 1	3 1 10
0 4	2 1 9
2 8	1 2 7
2 4	2 12
3 10	9 3 6
13 7	8 3 9
10 -3	7 3 12
0 0	6 2 12
3 3	5 2 12
1 3	4 2 12
3 1	3 1 11
1 1	2 1 11
4 3	1 1 11
1 4	0 1 10
4 1	-1 4 10
1 1	-2 7 10
5 4	3 0
0 6	4 1
2 3	2 2 2
3 0	5 2
1 3	4 1 1
6 6	2 2 2
1 3	6 1
3 3	2 1 3
4 2	
3 1	
1 1	
0 2	

I • Theta Puzzle

The *Theta Puzzle* consists of a base with 6 positions at the vertices of a regular hexagon and another position at the center, connected as shown in the figure below. There are six tokens labeled **A**, **B**, **C**, **D**, **E** and **F**. A single move of the puzzle is to move a token to an adjacent empty position (along an allowed connection – the line segments in the diagram below). The idea of the puzzle is to start with an initial arrangement of tokens with the center empty and, by a sequence of moves, get to the configuration in the figure below.
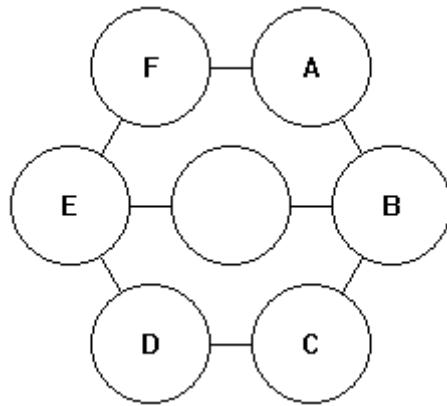
An initial position for the puzzle is given by a permutation of the letters **A** through **F**. The first letter starts at **A** in the figure, the next at **B** and so on.

A sequence of moves is specified by listing the labels of tokens to be moved, in the order they are to be moved.

For example, to solve the puzzle **FACDBE**, use the moves **BEFAB**.
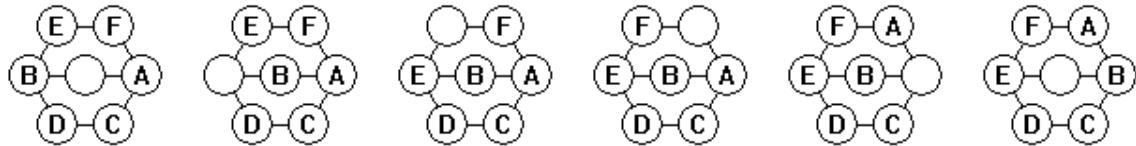
Note: Not all starting permutations can be solved.

Write a program which, given an initial permutation, either finds the *shortest* sequence of moves to solve the puzzle or determines that there is no solution.

Input

The first line of input contains a single integer **P**, (1 ≤ **P** ≤ 1000), which is the number of data sets that follow. Each data set is a single line that contains the data set number, followed by a space, followed by a permutation of the letters **A** through **F** giving the initial puzzle position.

Output

For each data set there is a single line of output. If there is no solution, the line contains a decimal integer giving the data set number followed by a single space, followed by the string **NO SOLUTION**. If there is a solution, the line contains a decimal integer giving the data set number followed by a single space, followed by the number of moves in the solution, followed by a single space, followed by the solution as a string of the letters **A** through **F**. If the number of moves is zero (0), you should still output the space after the 0, even though there is no string of letters.

Sample Input	Sample Output
12	1 5 BEFAB
1 FACDBE	2 0
2 ABCDEF	3 19 DABFECABFEDBACDEFAB
3 ADCEFB	4 NO SOLUTION
4 ADCEBF	5 29 BCDEBCAFBCAFBCEDFAECBAFDCBAFE
5 FEDCBA	6 NO SOLUTION
6 FEDCAB	7 19 CBFACBFACDEFACDEFAB
7 ECBFAD	8 NO SOLUTION
8 ECBFDA	9 13 CDAFBEDCBEDCB
9 DCEBFA	10 NO SOLUTION
10 DCEBAF	11 21 DAEBDAEBDCFEBDCABEFAB
11 CBEADF	12 16 FAEDBCAFBCAFEDCB
12 BDEAFC	