UVa Online Judge

110 Meta-Loopless Sorts

Sorting holds an important place in computer science. Analyzing and implementing various sorting algorithms forms an important part of the education of most computer scientists, and sorting accounts for a significant percentage of the world's computational resources. Sorting algorithms range from the bewilderingly popular Bubble sort, to Quicksort, to parallel sorting algorithms and sorting networks. In this problem you will be writing a program that creates a sorting program (a meta-sorter).

The problem is to create several programs whose output is a standard Pascal programs that sorts n numbers where n is the only input to the program you will write. The Pascal program generated by your program must have the following properties:

- They must begin with `program sort(input,output);`
- They must declare storage for exactly n `integer` variables. The names of the variables must come from the first n letters of the alphabet (a,b,c,d,e,f).
- A single `readln` statement must read in values for all the integer variables.
- Other than `writeln` statements, the only statements in the program are `if then else` statements. The boolean conditional for each `if` statement must consist of one strict inequality (either `<` or `>`) of two integer variables. Exactly $n!$ `writeln` statements must appear in the program.
- Exactly three semi-colons must appear in the programs
  1. after the program header: `program sort(input,output);`
  2. after the variable declaration: `... : integer;`
  3. after the `readln` statement: `readln(...);`
- No redundant comparisons of integer variables should be made. For example, during program execution, once it is determined that $a < b$, variables a and b should not be compared again.
- Every `writeln` statement must appear on a line by itself.
- The programs must compile. Executing the program with input consisting of any arrangement of any n distinct integer values should result in the input values being printed in sorted order.

For those unfamiliar with Pascal syntax, the example at the end of this problem completely defines the small subset of Pascal needed.

Input

The input consist on a number in the first line indicating the number M of programs to make, followed by a blank line. Then there are M test cases, each one consisting on a single integer n on a line by itself with $1 \leq n \leq 8$.

There will be a blank line between test cases.

Output

The output is M compilable standard Pascal programs meeting the criteria specified above.

Print a blank line between two consecutive programs.

Sample Input

```
1

3
```

Sample Output

```
program sort(input,output);
var
a,b,c : integer;
begin
  readln(a,b,c);
  if a < b then
    if b < c then
      writeln(a,b,c)
    else if a < c then
      writeln(a,c,b)
    else
      writeln(c,a,b)
  else
    if a < c then
      writeln(b,a,c)
    else if b < c then
      writeln(b,c,a)
    else
      writeln(c,b,a)
end.
```

UVa Online Judge

299 Train Swapping

At an old railway station, you may still encounter one of the last remaining "train swappers". A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains.

Once the carriages are arranged in the optimal order, all the train driver has to do, is drop the carriages off, one by one, at the stations for which the load is meant.

The title "train swapper" stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right.

The first train swapper had discovered that the bridge could be operated with at most two carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite direction, but train carriages can move either way, so who cares).

Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed, is a routine which decides for a given train the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create that routine.

Input

The input contains on the first line the number of test cases (N). Each test case consists of two input lines. The first line of a test case contains an integer L, determining the length of the train ($0 \leq L \leq 50$). The second line of a test case contains a permutation of the numbers 1 through L, indicating the current order of the carriages. The carriages should be ordered such that carriage 1 comes first, then 2, etc. with carriage L coming last.

Output

For each test case output the sentence: '`Optimal train swapping takes` S `swaps.`' where S is an integer.

Sample Input

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

Sample Output

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

UVa Online Judge

450 Little Black Book

The Public Relations Office is looking for an easier way to compile the information for their Black Book, a listing of all faculty members. Currently, each department types up a list of their faculty and submits the list to PR. PR then takes all the lists and makes a combined list that is sorted alphabetically by last name. So far they have been doing this task by hand, which takes far more time than the average PR employee has to spare!

What PR is looking for is a program that will take these faculty listings and combine them in a certain format that PR can use for inclusion in the Black Book. All the lists are stored in text files. Each department's file is sorted by last name. The program should take these individually sorted files and combine them into a single sorted file in the format shown below.

Input

The input file will contain, on the first line, a number (between 2 and 12) that reports the number of departments that your program should sort and write to the output file. Following the first line will be a series of sets of lines. The first line of each set contains a department title, and following lines contain the data to be sorted.

The information reported is in this order: Title, First Name, Last Name, Street Address, Home Phone, Work Phone, and Campus Mailbox. The information is delimited with commas.

A blank line separates two sets of lines. There are the same number of department titles and sets of data as the number on the first line reports.

Output

The format of each record held in the output file should be as follows:

```
----------------------------------------
< Title > < FirstName > < LastName >
< HomeAddress >
Department: < Department >
Home Phone: < HomePhone >
Work Phone: < WorkPhone >
Campus Box: < CampusBox >
```

The dashed line should be shown at the top of each record. The characters '<' and '>' show where a field should be placed. Please pay attention to spacing.

You may assume that all input files will be in the proper syntax, with no extra spaces. PR therefore expects that the data they are requesting will be in the proper syntax.

Sample Input

```
2
English Department
Dr.,Tom,Davis,Anystreet USA,555-2832,555-2423,823
Mrs.,Jessica,Lembeck,Center Street,555-2543,555-8584,928

Computer Science
Mr.,John,Euler,East Pleasure,555-1432,555-2343,126
```

Sample Output

```
----------------------------------------
Dr. Tom Davis
Anystreet USA
Department: English Department
Home Phone: 555-2832
Work Phone: 555-2423
Campus Box: 823
----------------------------------------
Mr. John Euler
East Pleasure
Department: Computer Science
Home Phone: 555-1432
Work Phone: 555-2343
Campus Box: 126
----------------------------------------
Mrs. Jessica Lembeck
Center Street
Department: English Department
Home Phone: 555-2543
Work Phone: 555-8584
Campus Box: 928
```

UVa Online Judge

612 DNA Sorting

One measure of "unsortedness" in a sequence is the number of pairs of entries that are out of order with respect to each other. For instance, in the letter sequence "`DAABEC`", this measure is 5, since `D` is greater than four letters to its right and `E` is greater than one letter to its right. This measure is called the number of inversions in the sequence. The sequence "`AACEDGG`" has only one inversion (`E` and `D`) — it is nearly sorted — while the sequence "`ZWQM`" has 6 inversions (it is as unsorted as can be — exactly the reverse of sorted).

You are responsible for cataloguing a sequence of DNA strings (sequences containing only the four letters `A`, `C`, `G`, and `T`). However, you want to catalog them, not in alphabetical order, but rather in order of "sortedness", from "most sorted" to "least sorted". All the strings are of the same length.

Input

The first line of the input is an integer M, then a blank line followed by M datasets. There is a blank line between datasets.

The first line of each dataset contains two integers: a positive integer n ($0 < n \leq 50$) giving the length of the strings; and a positive integer m ($0 < m \leq 100$) giving the number of strings. These are followed by m lines, each containing a string of length n.

Output

For each dataset, output the list of input strings, arranged from "most sorted" to "least sorted". If two or more strings are equally sorted, list them in the same order they are in the input file.

Print a blank line between consecutive test cases.

Sample Input

```
1

10 6
AACATGAAGG
TTTTGGCCAA
TTTGGCCAAA
GATCAGATTT
CCCGGGGGGA
ATCGATGCAT
```

Sample Output

```
CCCGGGGGGA
AACATGAAGG
GATCAGATTT
ATCGATGCAT
TTTTGGCCAA
TTTGGCCAAA
```

UVa Online Judge

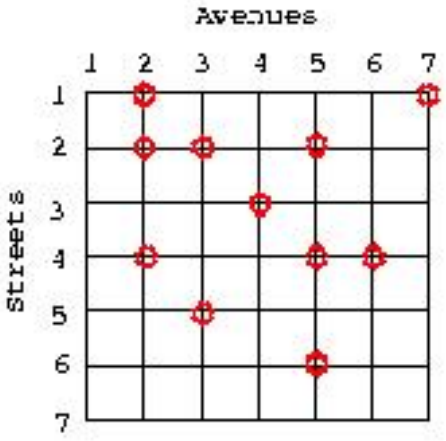
855 Lunch in Grid City

Grid city is a city carefully planned. Its street-map very much resembles that of downtown Manhattan in New York. Streets and avenues are orderly setup like a grid.

A group of friends living in Grid city decide to meet for lunch. Given that the group is dynamic, that is, its size may grow or shrink from time to time, they follow a written rule to determine the meeting point. It states that the meeting point is the one that minimizes the total distance the all group has to walk from their homes upto that point. If there is more than one candidate point, the rule imposes that the meeting point is the one corresponding to the smaller number for street and avenue. For large groups, this rule naturally avoids the usual long discussions that take place before aggreeing on a possible meeting point. For simplicity, consider that each person lives at a corner formed by a street and an avenue. You can also assume that the distance between two corners along one street or avenue is always one unit.

Your task is to suggest to the group their best meeting point (corner between a street and an avenue).

As an example, the following figure illustrates one such Crid city and the location of 11 friends. For this scenario, the best meeting point is street 3 and avenue 4. You can assume that streets and avenues are set and ordered as illustrated in this figure.



Please note that if we add another friend located at, say street 3 and avenue 5, making a total of 12 friends, then we would have two candidate meeting points, pairs (3,4) and (3,5). The rule clearly defines that street 3 and avenue 4 is the meeting point.

Given the size of a grid representing the Grid city and the locations of each person of the group of friends, your task is to determine the best meeting point following the rule of the group, as stated above.

Input

The first line of the input contains the number T of test cases, followed by T input blocks.

The first line of each test case consists of three positive numbers, the number of streets S, the number of avenues A (where $S \leq 1000$ and $A \leq 1000$), and the number of friends F (where $0 < F \leq 50000$). The following F input lines indicate the locations of the friends. A location is defined by two numbers, a street and an avenue, in this order.

Output

The output for each test case must list the best meeting point formatted as follows:

```
(Street: 3, Avenue: 4)
```

Each test case must be on a separate line.

Sample Input

```
2
2 2 2
```

```
1 1
2 2
7 7 11
1 2
1 7
2 2
2 3
2 5
3 4
4 2
4 5
4 6
5 3
6 5
```

Sample Output

```
(Street: 1, Avenue: 1)
(Street: 3, Avenue: 4)
```

UVa Online Judge

10107 What is the Median?

Median plays an important role in the world of statistics. By definition, it is a value which divides an array into two equal parts. In this problem you are to determine the current median of some long integers. Suppose, we have five numbers {1,3,6,2,7}. In this case, 3 is the median as it has exactly two numbers on its each side. {1,2} and {6,7}. If there are even number of values like {1,3,6,2,7,8}, only one value cannot split this array into equal two parts, so we consider the average of the middle values {3,6}. Thus, the median will be (3+6)/2 = 4.5. In this problem, you have to print only the integer part, not the fractional. As a result, according to this problem, the median will be 4 !

Input

The input file consists of series of integers X ($0 \leq X < 2^{31}$) and total number of integers N is less than 10000. The numbers may have leading or trailing spaces.

Output

For each input print the current value of the median.

Sample Input

```
1
3
4
60
70
50
2
```

Sample Output

```
1
2
3
3
4
27
4
```

UVa Online Judge

10327 Flip Sort

Sorting in computer science is an important part. Almost every problem can be solved effeciently if sorted data are found. There are some excellent sorting algorithm which has already acheived the lower bound $n \cdot \lg n$. In this problem we will also discuss about a new sorting approach. In this approach only one operation (Flip) is available and that is you can exchange two adjacent terms. If you think a while, you will see that it is always possible to sort a set of numbers in this way.

A set of integers will be given. Now using the above approach we want to sort the numbers in ascending order. You have to find out the minimum number of flips required. Such as to sort '1 2 3' we need no flip operation whether to sort '2 3 1' we need at least 2 flip operations.

Input

The input will start with a positive integer N ($N \leq 1000$). In next few lines there will be N integers.

Input will be terminated by EOF.

Output

For each data set print '`Minimum exchange operations :` M' where M is the minimum flip operations required to perform sorting. Use a seperate line for each case.

Sample Input

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

Sample Output

```
Minimum exchange operations : 0
Minimum exchange operations : 2
```

UVa Online Judge

10810 Ultra-QuickSort

In this problem, you have to analyze a particular sorting algorithm. The algorithm processes a sequence of n distinct integers by swapping two adjacent sequence elements until the sequence is sorted in ascending order. For the input sequence

`9 1 0 5 4` ,

Ultra-QuickSort produces the output

`0 1 4 5 9` .

Your task is to determine how many swap operations Ultra-QuickSort needs to perform in order to sort a given input sequence.

Input

The input contains several test cases. Every test case begins with a line that contains a single integer $n < 500,000$ — the length of the input sequence. Each of the the following n lines contains a single integer $0 \leq a[i] \leq 999,999,999$, the i-th input sequence element. Input is terminated by a sequence of length $n = 0$. This sequence must not be processed.

Output

For every input sequence, your program prints a single line containing an integer number *op*, the minimum number of swap operations necessary to sort the given input sequence.

Sample Input

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

Sample Output

```
6
0
```

UVa Online Judge

11462 Age Sort

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 ($0 < n \leq 2000000$), 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 ($\sim$ 25 MB) so use faster IO.

Sample Input

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

Sample Output

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

UVa Online Judge

11495 Bubbles and Buckets

Andrea, Carlos and Marcelo are close friends and spend their weekends by the swimming pool. While Andrea gets a suntan, both friends play *Bubbles*. Andrea, a very smart computer scientist, has already told them that she does not understand why they spend so much time playing a game so simple.

Using her laptop, Carlos and Marcelo generate a random integer N and a sequence, also random, which is a permutation from 1, 2, ..., N.

The game then begins. The players play by turns, and at each turn a player makes a move. Marcelo is always the first to play.

A move consists of choosing one pair of consecutive elements that are out of order in the sequence, and swapping both elements. For example, given the sequence 1, 5, 3, 4, 2, a player may swap 3 and 5 or 4 and 2, but cannot swap 3 and 4 nor 5 and 2. Continuing with the example, if the player decides to swap 5 and 3, the new sequence will be 1, 3, 5, 4, 2.

Sooner or later, the sequence will be sorted. The player that cannot make a move loses.

Andrea, with disdain, always says that it would be simpler to play Odd or Even, to the same effect. Your mission, in case you decide to accept it, is to determine who wins the game, given the initial permutation P.

Input

The input contains several test cases. Each test case is composed of a single line, in which all integers are separated by one space. Each line contains an integer N ($2 \leq N \leq 10^5$), followed by the initial sequence $P = (X_1, X_2, \ldots, X_N)$ of N distinct integers, with $1 \leq X_i \leq N$ for $1 \leq i \leq N$.

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line, containing the name of the winner, equal to '`Carlos`' or '`Marcelo`'.

Sample Input

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

Sample Output

```
Marcelo
Carlos
Carlos
Carlos
Carlos
Marcelo
```

UVa Online Judge

11714 Blind Sorting

I am a polar bear. But I am not just an ordinary polar bear. Yes I am extra ordinary! I love to play with numbers. One day my very good friend Mr. Panda came to me, and challenged me to solve a puzzle. He blindfolded me, and said that I have n distinct numbers. What I can ask is whether a-th number is larger than b-th number and he will answer me properly. What I have to do is to find out the largest and second largest number. I thought for a while and said "Come on, I will do it in minimum number of comparison."

Input

There will be a non-negative integer, n in each of the line of input where n is as described above. n will be less than any 10 digit prime number and not less than the smallest prime.

Output

For each n, output number of questions that I have to ask Mr. Panda in the worst case.

Sample Input

```
2
4
```

Sample Output

```
1
4
```

UVa Online Judge

11858 Frosh Week

During Frosh Week, students play various fun games to get to know each other and compete against other teams. In one such game, all the frosh on a team stand in a line, and are then asked to arrange themselves according to some criterion, such as their height, their birth date, or their student number. This rearrangement of the line must be accomplished only by successively swapping pairs of consecutive students. The team that finishes fastest wins. Thus, in order to win, you would like to minimize the number of swaps required.

Input

Input contains several test cases. For each test case, the first line of input contains one positive integer n, the number of students on the team, which will be no more than one million. The following n lines each contain one integer, the student number of each student on the team. No student number will appear more than once.

Output

For each test case, output a line containing the minimum number of swaps required to arrange the students in increasing order by student number.

Sample Input

```
3
3
1
2
```

Sample Output

```
2
```