Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

Dyzio is Jasiek's friend and he also likes riddles. Here is a riddle he came up with:

Jasiek, here is a piece of string, which has to be cut into smaller pieces. I will not tell you directly how to do it, but look at this sequence of zeros (0) and ones (1). A one at the begining means that the string has to be cut in half. If the first digit was zero, it would be the only digit in the sequence and mean you don't have to cut anything - I want the whole string. If you have to cut the string anyway then after the first 1 I wrote what to do with the left piece (according to the same rules as with the whole string) and then I wrote what to do with the right piece of string (all the time with the same rules of notation). Every time you have to cut the left piece first, only then can you cut the right one. Now start cutting and tell me, how many cuts you have to do until you have cut off the shortest piece.

Unfortunately mom hid the scissors from Jasiek, but luckily a computer was at hand and Jasiek quickly wrote a program simulating the string cutting. Can you write such a program?

Task

Write a program which

- reads (from standard input) description of the way the string is cut,
- counts how many cuts have to be made in order to get the first shortest piece.
- writes out the outcome (to standard output)

Input

Ten test cases (given one under another, you have to process all!). Each test case consists of two lines. In the first line there is a number *n (1<=n<=20000)*. In the second line one zero-one word (a sequence of zeros and ones without spaces between them) of length *n* - the description of the cutting procedure given by Dyzio.

Output

For every testcase your program should write (to the standard output) only one line with one integer equal to the number of cuts which have to be made in order to get the shortest piece.

Example

Input:
```
9
110011000
[and 9 test cases more]
```

Output:
```
4
[and 9 test cases more]
```

QUESTION – 2

An *n*-element permutation is an *n*-element sequence of distinct numbers from the set *{1, 2, ...,n}*. For example the sequence 2,1,4,5,3 is a 5-element permutation. We are interested in the longest increasing subsequences in a permutation. In this exemplary permutation they are of length 3 and there are exactly 2 such subsequences: 2,4,5 and 1,4,5. We will call a number belonging to any of the longest increasing subsequences a *supernumber*. In the permutation 2,1,4,5,3 the supernumbers are 1,2,4,5 and 3 is not a supernumber. Your task is to find all supernumbers for a given permutation.

Task

Write a program which

- reads a permutation from standard input,
- finds all its supernumbers,
- writes all found numbers to standard output.

Input

Ten test cases (given one under another, you have to process all!). Each test case consists of two lines. In the first line there is a number *n* (1<=n<=100000). In the second line: an *n*-element permutation - *n* numbers separated by single spaces.

Output

For every test case your program should write two lines. In the first line - the number of supernumbers in the input permutation. In the second line the supernumbers separated by single spaces in increasing order.

Example

Input:

```
5
2 1 4 5 3
[and 9 test cases more]
```

Output:

```
4
1 2 4 5
[and 9 test cases more]
```

QUESTION – 3

You are given:

- a positive integer n,
- an integer k, 1<=k<=n,
- an increasing sequence of k integers $0 < s_1 < s_2 < ... < s_k <= 2n$.

What is the number of proper bracket expressions of length 2n with opening brackets appearing in positions $s_1, s_2,...,s_k$?

Illustration

Several proper bracket expressions:

```
[[]][[[]][]]
[[[][]]][][[]]
```

An improper bracket expression:

```
[[[][]]][]][[]]
```

There is exactly one proper expression of length 8 with opening brackets in positions 2, 5 and 7.

Task

Write a program which for each data set from a sequence of several data sets:

- reads integers n, k and an increasing sequence of k integers from input,
- computes the number of proper bracket expressions of length 2n with opening brackets appearing at positions $s_1,s_2,...,s_k$,
- writes the result to output.

Input

The first line of the input file contains one integer d, 1 <= d <= 10, which is the number of data sets. The data sets follow. Each data set occupies two lines of the input file. The first line contains two integers n and k separated by single space, 1 <= n <= 19, 1 <= k <= n. The second line contains an increasing sequence of k integers from the interval [1;2n] separated by single spaces.

Output

The i-th line of output should contain one integer - the number of proper bracket expressions of length 2n with opening brackets appearing at positions $s_1, s_2,...,s_k$.

Example

Sample input:

```
5
1 1
1
1 1
2
2 1
1
3 1
2
4 2
5 7
```

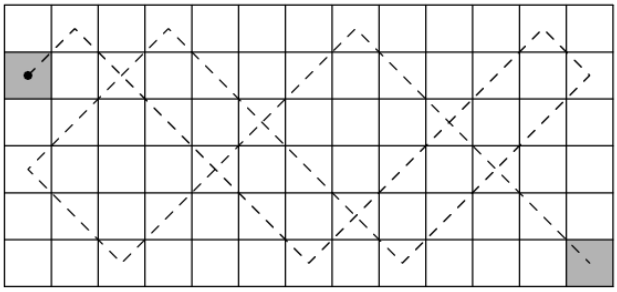
Sample output:

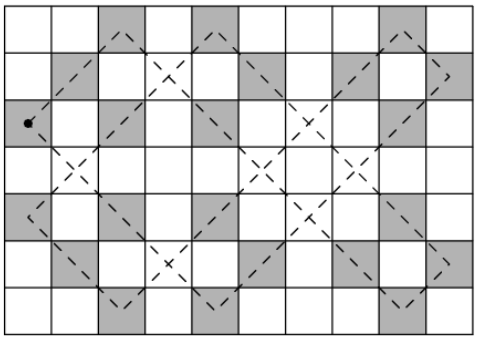
```
1
0
2
3
2
```

QUESTION – 4

On the rectangular chessboard of n x m square fields we choose one field adjacent to the edge of the chessboard, called the starting field. Then we put a ball in the center of this field and push it to roll through the chessboard. The diameter of the ball equals the width (and height) of chessboard field. The angle between the direction of ball movement and the edge of the chessboard equals 45 degrees. The ball bounces off the edges of the chessboard: if the ball touches the edge of the chessboard then each composite of its velocity perpendicular to the edge touched is reversed. At the start the ball is pushed toward increasing coordinates (when the starting field is a field of the highest coordinate, the ball bounces momentarily).

We assign a point to a field of the chessboard each time the point of adjacency between the ball and the chessboard enters the interior of the field. The game is over when a point is assigned to the starting field. What is the number of fields to which an odd number of points is assigned? The following figures illustrate the problem. The route of the ball is marked with a dashed line. Fields with the odd number of points are shadowed.





Task

Write a program which for each data set from a sequence of several data sets:

- reads the dimensions of the chessboard and the coordinates of starting field from input,
- computes the number of fields with the odd number of points,
- writes the result to output.

Input

The first line of the input file contains one integer d, 1 <= d <= 10, which is the number of data sets. The data sets follow. Each data set occupies one line of the input file. Such a line consists of four integers x, y, a, b separated with single spaces. These integers are the x- and y-dimensions of the chessboard and x- and y-coordinates of the starting field, respectively. Integers x and y are greater than two, the number of fields of the chessboard does not exceed 10^9,the starting field is adjacent to the edge of the chessboard.

Output

The i-th line of output should contain one integer which is equal to the number of fields of the chessboard with the odd number of points.

Example

```
Sample input:

2
13 6 1 5
10 7 1 5

Sample output:

2
22
```

QUESTION – 5

The old tube screen to your computer turned out to be the cause of your chronic headaches. You therefore decide to buy one of these new flat TFT monitors. At the entrance of the computer shop you see that it is quite full with customers.

In fact, the shop is rather packed with customers and moving inside involves a certain amount of elbowing. Since you want to return home quickly to complete your half finished SPOJ tasks, you want to sidestep the crowd as much as possible. You examine the situation somewhat closer and realise that the crowding is less in some parts of the shop. Thus, there is reason for hope that you can reach your goal in due time, provided that you take the shortest way. But which way is the shortest way?

You sketch the situation on a piece of paper but even so, it is still a tricky affair. You take out your notebook from your pocket and start to write a program which will find the shortest way for you.

Input

The first line of the input specifies the width w and height h of the shop. Neither dimension exceeds 25.

The following h lines contain w characters each. A letter X symbolises a shelf, the letter S marks your starting position, and the letter D marks the destination (i.e. the square in front of the monitors). All free squares are marked with a digit from 1 to 9, meaning the number of seconds needed to pass this square.

There are many test cases separated by an empty line. Input terminates with width and height equal 0 0.

Output

Your program is to output the minimum number of seconds needed to reach to destination square. Each test case in a separate line. Movements can only be vertical and horizontal. Of course, all movements must take place inside the grid. There will always be a way to reach the destination.

Example

Sample input:

```
4 3
X1S3
42X4
X1D2
```

```
5 5
S5213
2X2X5
51248
4X4X2
1445D

0 0
```

Sample output:

```
4
23
```