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

Another chapter of the municipal chronicles of a well known unbelievable lordly major town (if this town is not well known to you, you might want to solve problem CSTREET first) tells us the following story:
Once upon a time the citizens of the unbelievable lordly major town decided to elect a major. At that time this was a very new idea and election campaigns were completely unknown. But of course several citizens wanted to become major and it didn't took long for them to find out, that promising nice things that never will become real tends to be useful in such a situation. One candidate to be elected as a major was Ivo sometimes called the benefactor because of his valuable gifts to the unbelievably lordly major towns citizens.
One day before the election day Ivo the benefactor made a promise to the citizens of the town. In case of his victory in the elections he would ensure that on one of the paved streets of the town street lamps would be erected and that he would pay that with his own money. As thrifty as the citizens of the unbelievable lordly major town were, they elected him and one day after the elections they presented him their decision which street should have street lamps. Of course they chose not only the longest of all streets but renamed several streets so that a very long street in the town existed.
Can you find how long this street was? To be more specific, the situation is as follows. You are presented a list of all paved streets in the unbelievable lordly major town. As you might remember from problem CSTREET in the town the streets are paved in a way that between every two points of interest in the town exactly one paved connection exists. Your task is to find the longest distance that exists between any two places of interest in the city.

Input

The first line of input contains the number of testcases t.
The first line of each testcase contains the number of places (2 <=n<=50000) in the town. Each street is given at one line by two places (1<=a,b<=n) and the length of the street (0<=l<20000).

Output

For each testcase output one line which contains the maximum length of the longest street in the city.

Example

Input:

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

Output:

```
12
```

QUESTION – 2

Farmer John has built a new long barn, with N (2 <= N <= 100,000) stalls. The stalls are located along a straight line at positions x1,...,xN (0 <= xi <= 1,000,000,000).

His C (2 <= C <= N) cows don't like this barn layout and become aggressive towards each other once put into a stall. To prevent the cows from hurting each other, FJ want to assign the cows to the stalls, such that the minimum distance between any two of them is as large as possible. What is the largest minimum distance?

Input

t – the number of test cases, then *t* test cases follows.
* Line 1: Two space-separated integers: N and C
* Lines 2..N+1: Line i+1 contains an integer stall location, xi

Output

For each test case output one integer: the largest minimum distance.

Example

Input:

```
1
5 3
1
2
8
4
9
```

Output:

```
3
```

QUESTION – 3

Bob has a difficult job. He must distribute advertising booklets for extra school activities in different schools. The booklets have different number of pages. Bob has a list with the number of pages of each booklet and the number of schools that he must visit. He has to distribute the booklets such that each school gets a number of booklets equal to either the lower integer part (LIP),or the upper integer part (UIP) of the number of booklets divided by the number of

schools. Poor Bob must obey other rules too. He must distribute all the **UIP** number of booklets first and then the **LIP** number of booklets.

Any booklet **A** that is distributed to a school **S_i** must have fewer or at most an equal number of pages that any other booklet B that is distributed to a school **S_j**, if **S_i** gets the booklets before **S_j** (i.e if **i** < **j** then **pages(A)** <= **pages(B)**). When Bob distributes the booklets to a school he must distribute them in the same relative order in which they are on his list.

Moreover, he must distribute them very fast. When he comes back to the advertising company his boss verifies if he accomplished well his task, by asking him the number of pages of the first booklet distributed to a specific school, following the order in which Bob visited the schools (starting with 0). Difficult job, isn't it? Can you help him?

Input

The input starts with a line containing a single integer **t** <= 20, the number of test cases. **t** test cases follow.

Each data set in the input stands for a particular set of bookets. For each set of booklets the input contains the number of schools, the school specified by Bob's boss, the number of booklets (**less than 3000**), the number of pages of each booklet (fits in integer). White spaces can occur freely between the numbers in the input. The input data are correct.

Output

For each set of data the program prints the result to the standard output on a separate line. The solution is represented by the number of pages of the first booklet distributed to the specified school.

Example

Input:

```
1
3
2
7
3 5 9 1 11 14 2
```

Output:

```
11
```

QUESTION – 4

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is "There's no place like home on a snowy night" and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character 'x' to pad the message out to make a rectangle, although he could have used any letter. Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2...20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Example

Input:

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Output:

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

QUESTION – 5

"Hike on a Graph" is a game that is played on a board on which an undirected graph is drawn. The graph is complete and has all loops, i.e. for any two locations there is exactly one arrow between them. The arrows are coloured. There are three players, and each of them has a piece. At the beginning of the game, the three pieces are in fixed locations on the graph. In turn, the players may do a move. A move consists of moving one's own piece along an arrow to a new location on the board. The following constraint is imposed on this: the piece may only be moved along arrows of the same colour as the arrow between the two opponents' pieces.

In the sixties ("make love not war") a one-person variant of the game emerged. In this variant one person moves all the three pieces, not necessarily one after the other, but of course only one at a time. Goal of this game is to get all pieces onto the same location, using as few moves as possible. Find out the smallest number of moves that is necessary to get all three pieces onto the same location, for a given board layout and starting positions.

Input

The input file contains several test cases. Each test case starts with the number *n*. Input is terminated by *n=0*. Otherwise, $1<=n<=50$. Then follow three integers p_1, p_2, p_3 with $1<=p_i<=n$ denoting the starting locations of the game pieces. The colours of the arrows are given next as a $n \times n$ matrix *m* of whitespace-separated lower-case letters. The element m_{ij} denotes the colour of the arrow between the locations *i* and *j*. Since the graph is undirected, you can assume the matrix to be symmetrical.

Output

For each test case output on a single line the minimum number of moves required to get all three pieces onto the same location, or the word "impossible" if that is not possible for the given board and starting locations.

Example

Input:

```
3 1 2 3
r b r
b b b
r b r
2 1 2 2
y g
g y
```

0

Output:

2
impossible