



## SOFTEC 2018

---

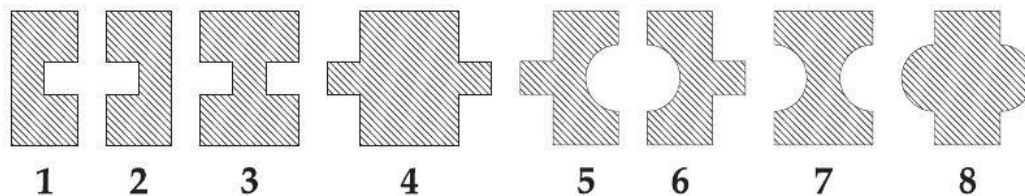
### Programming Competition Rules

- Allowed programming languages are Java, (MS Visual & GNU) C++ and Python3. Support for any other programming language will not be provided.
- The duration of this competition round is 3 hours.
- The competition would include a number of challenging problems each of which would require you to develop some algorithm and get the output in a fixed format on the console. The output would be matched **character- to-character** with the output of our program through a PC.
- The format must be exactly same as mentioned in the question paper. Any differences from defined format will be considered a wrong answer.
- For Java, don't use any package statement. Usage will be considered incorrect answer.
- For Python, don't try to use or download any external libraries since only core python features are allowed.
- Your program must terminate immediately after completion. Do not use any statement to hold termination e.g system("pause"), getch etc
- The scoring would hence be Yes/No i.e. either correct or incorrect.
- The decision of the Judges will be final and if any participant keeps arguing with the host team on this matter, his/her's team will be disqualified.
- Usage of internet is strictly prohibited. Usage of internet will result in immediate disqualification
- One team is only allowed to use one system. Usage of multiple systems will not be allowed.
- If there is a tie on the number of problems solved, the scoring will depend on the **time taken and total incorrect submissions made** since the start of competition to the time of correct submission. (This is automatically judged the PC and viewable on board).
- Live competition boards will be available on each terminal, but it will be disabled in the last half hour of competition



- Use of mobile phones and other communication devices is strictly prohibited during the competition. Anyone seen using devices will be immediately disqualified. Such devices should be switched off and placed in pockets / hand bags and not visible or accessible during the competition.
- No external edibles are allowed inside the lab. Please be careful with the drinkables near electronic hardware.
- Once the competition starts, any discussion between two different teams may result in immediate disqualification from the competition.
- The test cases on the judge's side may not be the same as given with the problem. You have to cater for number of test cases, end of file and other restrictions if they're not given on the first line of test cases file.
- Decision of the competition judges shall be considered as final and cannot be challenged.
- Teams creating disturbance will be immediately disqualified.
- Disqualification from competition also means escorting out of competition venue and a note will be sent to their respective institutes.

# Problem1: Blocks

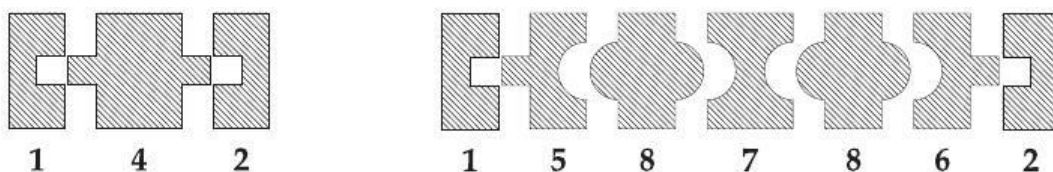
Blocks is a game where you're given wooden pieces that come in eight shapes as shown below:



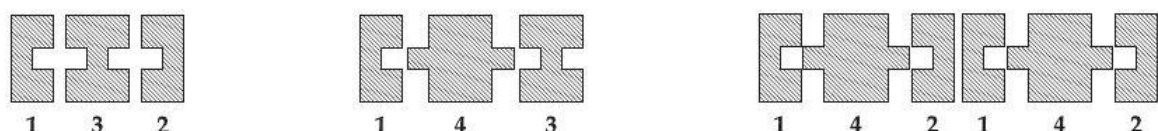
The objective of the game is to assemble the widest rectangle that can be made of a subset of the given pieces with the following conditions:

1. The rectangle must have smooth edges. In other words, the left-most piece must be  (piece #1) and the right-most piece must be  (piece #2.)
2. Adjacent pieces must interlock properly. For example, either piece #4 or piece #5 must appear to the right of piece #1. Similarly, piece #4 can appear to the right of piece #1 or piece #3.
3. No piece interlocks to the left of piece #1. No piece interlocks to the right of piece #2.
4. For each piece #1, the rectangle must have a matching piece #2. Similarly, for each piece #5, there must be a matching piece #6.

For example, the following two examples are valid arrangements:



Whereas the following three are not:



A computer company is interested in building a Blocks video game and has hired you to write a program that determines if a given pieces' arrangement is valid according to the rules above, or not.

### **Input**

Your program will be tested on one or more test cases. Each test case is specified on a separate input line. Each piece is specified using the digit associated with it as in the previous figure. An arrangement is specified by listing its digits with no spaces between the digits. Each arrangement will have at least one piece, but no more than 10; 000 pieces.

The last line in the input file will have a single '0'. That line is not part of the test cases.

### **Output**

For each test case, output the result on a single line using the following format:

*k. result*

Where *k* is the test case number (starting at 1,) and result is 'VALID' if the arrangement is valid, or 'NOT' if it's not.

### **Sample Input**

142

1587862

132

143

0

### **Sample Output**

1. VALID

2. VALID

3. NOT

4. NOT

## Problem 2: Letter Stamper

Roland is a high-school math teacher. Every day, he gets hundreds of papers from his students. For each paper, he carefully chooses a letter grade: 'A', 'B' or 'C'. (Roland's students are too smart to get lower grades like a 'D' or an 'F'). Once the grades are all decided, Roland passes the papers onto his assistant - you. Your job is to stamp the correct grade onto each paper.

You have a low-tech but functional letter stamp that you use for this. To print out a letter, you attach a special plate to the front of the stamp corresponding to that letter, dip it in ink, and then apply it to the paper.

The interesting thing is that instead of removing the plate when you want to switch letters, you can just put a new plate on top of the old one. In fact, you can think of the plates on the letter stamp as being a stack, supporting the following operations:

- Push a letter on to the top of the stack. (This corresponds to attaching a new plate to the front of the stamp.)
- Pop a letter from the top of the stack. (This corresponds to removing the plate from the front of the stamp.)
- Print the letter on the top of the stack. (This corresponds to actually using the stamp.) Of course, the stack must actually have a letter on it for this to work.

Given a sequence of letter grades ('A', 'B', and 'C'), how many operations do you need to print the whole sequence in order? The stack begins empty, and you must empty it when you are done. However, you have unlimited supplies of each kind of plate that you can use in the meantime.

For example, if you wanted to print the sequence "ABCCBA", then you could do it in 12 operations, as shown below:

Operation	Printed so far	Stack
0. -	-	-
1. Push A	-	A
2. Print	A	A
3. Push B	A	AB
4. Print	AB	AB
5. Push C	AB	ABC
6. Print	ABC	ABC
7. Print	ABCC	ABC
8. Pop	ABCC	AB
9. Print	ABCCB	AB
10. Pop	ABCCB	A
11. Print	ABCCBA	A
12. Pop	ABCCBA	-

## Input

The first line of the input file contains the number of cases, **T**. **T** test cases follow, one per line. Each of these lines contains a single string **S**, representing the sequence of characters that you want to print out in order.

## Output

For each test case, output one line containing "Case #x: N", where x is the case number (starting from 1) and N is the minimum number of stack operations required to print out **S**.

## Limits

**S** is a non-empty string containing only the letters 'A', 'B', and 'C'.

## Dataset

$1 \leq T \leq 100$ .

**S** has at most 100 characters.

Input	Output
2	Case #1: 12
ABCCBA	Case #2: 13
AAABAAB	

## Problem 3: Magic Trick

Recently you went to a magic show. You were very impressed by one of the tricks, so you decided to try to figure out the secret behind it!

The magician starts by arranging 16 cards in a square grid: 4 rows of cards, with 4 cards in each row. Each card has a different number from 1 to 16 written on the side that is showing. Next, the magician asks a volunteer to choose a card, and to tell him which row that card is in.

Finally, the magician arranges the 16 cards in a square grid again, possibly in a different order. Once again, he asks the volunteer which row her card is in. With only the answers to these two questions, the magician then correctly determines which card the volunteer chose. Amazing, right?

You decide to write a program to help you understand the magician's technique. The program will be given the two arrangements of the cards, and the volunteer's answers to the two questions: the row number of the selected card in the first arrangement, and the row number of the selected card in the second arrangement. The rows are numbered 1 to 4 from top to bottom.

Your program should determine which card the volunteer chose; or if there is more than one card the volunteer might have chosen (the magician did a bad job); or if there's no card consistent with the volunteer's answers (the volunteer cheated).

### Solving this problem

Usually, Google Code Jam problems have 1 Small input and 1 Large input. This problem has only **1 Small input**. Once you have solved the Small input, you have finished solving this problem.

### Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing an integer: the answer to the first question. The next 4 lines represent the first arrangement of the cards: each contains 4 integers, separated by a single space. The next line contains the answer to the second question, and the following four lines contain the second arrangement in the same format.

## Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1).

If there is a single card the volunteer could have chosen, y should be the number on the card. If there are multiple cards the volunteer could have chosen, y should be "Bad magician!", without the quotes. If there are no cards consistent with the volunteer's answers, y should be "Volunteer cheated!", without the quotes. The text needs to be exactly right, so consider copying/pasting it from here.

## Limits

$1 \leq T \leq 100$ .

$1 \leq \text{both answers} \leq 4$ .

Each number from 1 to 16 will appear exactly once in each arrangement.

## Sample

Input	Output
3	Case #1: 7
2	Case #2: Bad magician!
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Case #3: Volunteer cheated!
3	
1 2 5 4 3 11 6 15 9 10 7 12 13 14 8 16	
2	
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	
2	
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	
2	
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	
3	
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	



# Problem 4: Buffed Buffet

You are buying lunch at a buffet. A number of different dishes are available, and you can mix and match them to your heart's desire. Some of the dishes, such as dumplings and roasted potatoes, consist of pieces of roughly equal size, and you can pick an integral number of such pieces (no splitting is allowed). Refer to these as "discrete dishes." Other dishes, such as tzatziki or mashed potatoes, are fluid and you can pick an arbitrary real-valued amount of them. Refer to this second type as "continuous dishes."

Of course, you like some of the dishes more than others, but how much you like a dish also depends on how much of it you have already eaten. For instance, even if you generally prefer dumplings to potatoes, you might prefer a potato over a dumpling if you have already eaten ten dumplings. To model this, each dish  $i$  has an initial tastiness  $t_i$ , and a rate of decay of the tastiness  $\Delta t_i$ . For discrete dishes, the tastiness you experience when eating the  $n^{\text{th}}$  item of the dish is  $t_i - (n - 1)\Delta t_i$ . For continuous dishes, the tastiness you experience when eating an infinitesimal amount  $dx$  grams of the dish after already having eaten  $x$  grams is  $(t_i - x\Delta t_i)dx$ . In other words, the respective total amounts of tastiness you experience when eating  $N$  items of a discrete dish or  $X$  grams of a continuous dish are as follows:

$$\sum_{n=1}^N (t_i - (n - 1)\Delta t_i) \quad \text{and} \quad \int_0^X (t_i - x\Delta t_i)dx$$

For simplicity, do not take into account that different dishes may or may not go well together, so define the total tastiness that you experience from a meal as the sum of the total tastinesses of the individual dishes in the meal (and the same goes for the weight of a meal – there are no food antiparticles in the buffet!).

You have spent days of painstaking research determining the numbers  $t_i$  and  $\Delta t_i$  for each of the dishes in the buffet. All that remains is to compute the maximum possible total tastiness that can be achieved in a meal of weight  $w$ . Better hurry up, lunch is going to be served soon!

## Input

The input consists of a single test case. The first line of input consists of two integers  $d$  and  $w$  ( $1 \leq d \leq 250$  and  $1 \leq w \leq 10000$ ), where  $d$  is the number of different dishes at the buffet and  $w$  is the desired total weight of your meal in grams.

Then follow  $d$  lines, the  $i^{\text{th}}$  of which describes the  $i^{\text{th}}$  dish. Each dish description is in one of the following two forms:

- A description of the form "D  $w_i t_i \Delta t_i$ " indicates that this is a discrete dish where each item weighs  $w_i$  grams, with initial tastiness  $t_i$  and decay of tastiness  $\Delta t_i$ .
- A description of the form "C  $t_i \Delta t_i$ " indicates that this is a continuous dish with initial tastiness  $t_i$  and decay of tastiness  $\Delta t_i$ .

The numbers  $w_i$ ,  $t_i$ , and  $\Delta t_i$  are integers satisfying  $1 \leq w_i \leq 10000$  and  $0 \leq t_i \Delta t_i \leq 10000$ .

## Output

Display the maximum possible total tastiness of a meal of weight  $w$  based on the available dishes. Give the answer rounded off to 5 decimal places. If it is impossible to make a meal of total weight exactly  $w$  based on the available dishes, display impossible.

### Sample Input 1

```
2 15
D 4 10 1
C 6 1
```

### Sample Output 1

```
40.50000
```

### Sample Input 2

```
3 15
D 4 10 1
C 6 1
C 9 3
```

### Sample Output 2

```
49.00000
```

### Sample Input 3

```
2 19
D 4 5 1
D 6 3 2
```

### Sample Output 3

```
impossible
```

## Problem 5: Goro Sort

Goro has 4 arms. Goro is very strong. You don't mess with Goro. Goro needs to sort an array of  $N$  different integers. Algorithms are not Goro's strength; strength is Goro's strength. Goro's plan is to use the fingers on two of his hands to hold down several elements of the array and hit the table with his third and fourth fists as hard as possible. This will make the unsecured elements of the array fly up into the air, get shuffled randomly, and fall back down into the empty array locations.

Goro wants to sort the array as quickly as possible. How many hits will it take Goro to sort the given array, on average, if he acts intelligently when choosing which elements of the array to hold down before each hit of the table? Goro has an infinite number of fingers on the two hands he uses to hold down the array.

More precisely, before each hit, Goro may choose any subset of the elements of the array to freeze in place. He may choose differently depending on the outcomes of previous hits. Each hit permutes the unfrozen elements uniformly at random. Each permutation is equally likely.

### Input

The first line of the input gives the number of test cases,  $T$ .  $T$  test cases follow. Each one will consist of two lines. The first line will give the number  $N$ . The second line will list the  $N$  elements of the array in their initial order.

### Output

For each test case, output one line containing "Case # $x$ :  $y$ ", where  $x$  is the case number (starting from 1) and  $y$  is the expected number of hit-the-table operations when following the best hold-down strategy. Answers with an absolute or relative error of at most  $10^{-6}$  will be considered correct.

### Limits

$1 \leq T \leq 100$ ;

The second line of each test case will contain a permutation of the  $N$  smallest positive integers.

$1 \leq N \leq 10$ ;

## Sample

Input	Output
3	Case #1: 2.000000
2	Case #2: 2.000000
2 1	Case #3: 4.000000
3	
1 3 2	
4	
2 1 4 3	

## Explanation

In test case #3, one possible strategy is to hold down the two leftmost elements first. Elements 3 and 4 will be free to move. After a table hit, they will land in the correct order [3, 4] with probability  $1/2$  and in the wrong order [4, 3] with probability  $1/2$ . Therefore, on average it will take 2 hits to arrange them in the correct order. After that, Goro can hold down elements 3 and 4 and hit the table until 1 and 2 land in the correct order, which will take another 2 hits, on average. The total is then  $2 + 2 = 4$  hits.

## Problem 6: Dire Straights

You are playing a card game, where each card has an integer number written on it.

To play the game, you are given some cards — your *hand*. Then you arrange the cards in your hand into *straights*. A straight is a set of cards with consecutive values; e.g. the three cards {3, 4, 5}, or the single card {7}. You then receive a number of dollars equal to the length of the shortest straight. If you have no cards, you can form no straights, so you get zero dollars.

You will be given a series of test cases, each of which describes the cards you will have in your hand. Find the maximum number of dollars you can receive for each test case.

### Input

The first line of the input contains the number of test cases, **T**. Each test case consists of one line. Each line contains **N**, the number of cards in your hand, followed by **N** integers giving the numbers on those cards. These numbers are all space-separated.

### Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the maximum number of dollars you can receive.

### Limits

$$1 \leq T \leq 100$$

The numbers on the cards are between 1 and 10000.

$$0 \leq N \leq 10$$

### Sample

Input	Output
4	Case #1: 10
10 1 2 3 4 5 10 9 8 7 6	Case #2: 4
8 101 102 103 104 105 106 103 104	Case #3: 0
0	Case #4: 1
5 1 2 3 4 9	

In case 1, you have ten cards numbered 1 to 10, so you make one straight of length 10, and get 10 dollars.

In case 2, you could make two straights {101,102,103,104,105,106} and {103,104} and get 2 dollars. But it would be better to make {101,102,103,104} and {103,104,105,106} and get 4 dollars.

In case 4, the card with the number 9 must be in a straight containing only that card. So you get 1 dollar.

In case 3, you have zero cards, so you get zero dollars. You don't get money for nothing.