13081 XOR Sequence Revisited

Jerry loves XOR sequence. He has an array A. The array is described below:

- $A_0 = 1$
- $A_x = A_{x-1} \oplus x$ for $x > 0$ ($\oplus$ is symbol of XOR)

First few elements of the array are [1, 0, 2, 1, 5, 0, 6, 1, 9].

Given a range $[L, R]$, find the **AND** of all the elements between A_L and A_R (inclusive), i.e. You need to find $A_L \& A_{L+1} \& A_{L+2} \& \ldots \& A_R$ where & is the symbol of bitwise **AND** operator.

Input

First line will contain an integer number T ($1 \leq T \leq 100000$), denoting number of test cases. Each of the next T lines contains one test case. Each test case will contain two integers L and R ($0 \leq L \leq R \leq 10^{15}$).

Output

For each case, print the answer in a single line.

Sample Input

```
2
2 4
2 2
```

Sample Output

```
0
2
```

13082 High School Assembly

It's Saturday morning of a new week and students of National High School have gathered on the central ground for an assembly.

At the beginning, students from different classes stand in their own lines. Then the class teachers of each class move over from front to back and organize the line according to the increasing order of students' heights. They can pick a student from any position and send him to the end of the line.

Mr. Kapono Khan is the class teacher of 7th grade. He doesn't like this job of walking along the line back and forth. So he wants to organize the students with minimal number of moves. As usual, you are here to help Mr. Khan. Given the heights of each student of his class, your job is to find out minimum number of moves required to sort the students based on the increasing order of their heights. Picking up a student from any position and sending him to the end is **defined as a move** for this problem. Luckily students of his class have unique heights.

Input

There will be T test cases, ($T \leq 100$).

Input for each case will start with an integer, n ($1 \leq n \leq 10^4$) which represents the number of students of the class. Then an array of n integers will follow where $1 \leq H_i \leq n$ and each height is unique.

Output

For each case, print case number using the format '`Case` x`: `' (without quotes) followed by an integer showing minimum number of moves required to sort the line in increasing order.

Explanation of Sample I/O

Case 1: move 3 to last, move 4 to last, move 5 to last.

Sample Input

```
2
5
5 1 3 2 4
9
4 5 1 2 6 3 8 9 7
```

Sample Output

```
Case 1: 3
Case 2: 6
```

13083 Yet another GCDSUM

Given the value of N, you will have to find the value of S. The definition of S is given in the following code:

```
S=0;
for(i=1;i<=N;i++)
  for(j=1;j<=N;j++)
    if((N \% i)==0 && (N \% j)==0)
        S+=gcd(i,j);
```

/*Here ‘`gcd()`’ is a function that finds the greatest common divisor of the two input numbers. ‘`%`’ is standard remainder sign from C/C++/java syntax where ‘a `%` b’ is the remainder of a modulo b, so ‘`(`N `%` i`)== 0 && (`N `%` j`) == 0`’ means N is divisible by both i and j*/

Input

First line of the input is T ($T \leq 100$), then T test cases follows in next T lines. Each line contains an integer N ($1 \leq N \leq 100000000000000$ or 10^{14}). The meaning of N is given in the problem statement.

Output

For each test case print a line in ‘`Case` I`:` S’ format where I is case number and S is the value for the N of this case. The value of S will fit in a **64-bit** signed integer.

Sample Input

```
12
1
2
3
4
5
6
7
8
9
10
1000
10000
```

Sample Output

```
Case 1: 1
Case 2: 5
Case 3: 6
Case 4: 15
Case 5: 8
Case 6: 30
Case 7: 10
```

```
Case 8: 37
Case 9: 23
Case 10: 40
Case 11: 8584
Case 12: 97027
```

13084 Cameras as Invigilators

It is now 2040 AD. Gone are the days when thousands of teachers spent their precious time in exam halls as invigilators. Now exams are conducted in large square shaped rooms and two high precision cameras are placed in two corners of that room. Such a (6×6) exam room is shown on the left and this room can accommodate only (7×7) 49 students. But as there are cameras on the two upper corners so actually there is sitting arrangements for $49 - 2 = 47$ students. But actual exam rooms can be square shaped and very large. So in general an $(n \times n)$ exam room has sitting arrangement for students. For this problem, n can be as large as 5000. As the room is very large and the students sit in grid pattern and far away from one another so the students and two cameras can be considered as points. The two cameras are high precision cameras and they can pin point each student very accurately. If the angular distance between two points is zero the camera considers that they are at the same line with the camera, but here lies the biggest bug of the camera which will be explained below.

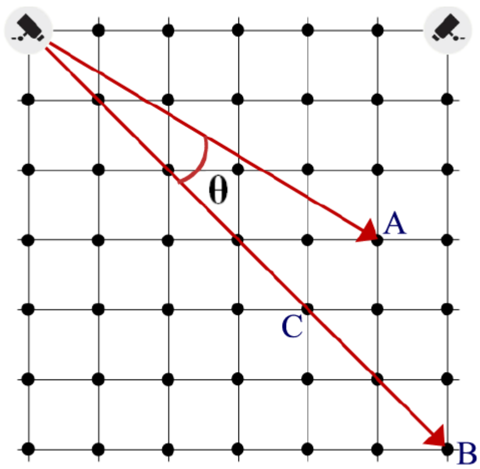


Figure 1:

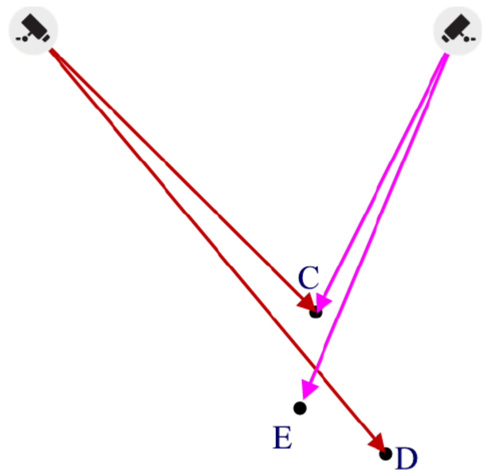


Figure 2:

The camera can reasonably accurately decide whether two points are at the same line with respect to the camera. For example, in the picture on the left, point B, C and the camera at the upper left corner (Camera 1) is at the same line and the camera can detect it. Point A and B makes an angle θ ($\theta >> 0$) with this camera so A B and C are not at the same line. But if the value of is very small then the camera can wrongly think that A, B and the Camera are at the same line and that is when this Camera will not function well as an invigilator. Then these two points will become troublesome points. For this problem θ is expressed as $tan^{-1}(\frac{1}{k})$, here $n^2 \leq k \leq 2n^2$. The range of k is such because when the room is large the exam authority can afford to buy higher precision cameras (Or cameras with more accuracy). To reduce the number of troublesome points, a 2nd Camera is installed at the upper right corner (Camera 2). Now a point that is troublesome for Camera 1 may not be troublesome for Camera 2. But still there may be some points which are troublesome for both the cameras. The figure on the right can be used to explain this scenario. We can see that point C and D makes a very small angle with Camera 1. So C and D may be considered troublesome for Camera 1. On the other hand, point C and E makes very small angle with Camera 2. So these two can be considered troublesome with respect to Camera 2. So point C is troublesome with respect to both the cameras. Given the size of the examination hall and value of k your job is to find out the number points or locations that are troublesome for both the cameras.

Input

First line of the input file contains a positive integer T ($T \leq 10$) which denotes the number of lines to follow.

Each following line contains two integers n ($10 \leq n \leq 5000$) and k ($n^2 \leq k \leq 2n^2$). That means you have to consider an examination hall that can be represented as $(n \times n)$ grid, and there are two cameras (one at the upper left corner and the other at the upper right corner) and examinees sit on all other lattice points that are not outside the exam hall. And both the cameras have the defect that if two points or locations have angular distance less than $tan^{-1}(\frac{1}{k})$ with respect to the camera, it considers them collinear with the camera and both of them becomes troublesome points.

Output

For each test case you should produce one line of output which contains an integer T which denotes the total number of points that are troublesome with respect to both the cameras.

Sample Input

```
2
100 10000
1000 1500000
```

Sample Output

```
30
0
```

13085 Forming Teams

You are currently in charge of a large multinational company. You have many projects in hand. There are N employees currently in your company and all of them have a unique ID, numbered from 1 to N. You want to form a non-empty set of teams of equal size, from these employees, such that each team works under a unique project and each employee works in exactly one team. Your task is to find the number of ways to form such sets of teams.

Two ways are different, if the number or size of the teams are different, or if a pair of employees works in the same team in one formation, but works in different teams in another formation.

Input

The first line of each input contains a single integer T ($1 \leq T \leq 5000$), which denotes the number of test cases.

The next T lines contain a single integer N ($1 \leq N \leq 10^6$).

Output

For each test case, output the case number, followed by the number of ways to form non-empty sets of equal sized teams from N employees. Since the result can be large, print it *modulo* 1000000007.

See the sample input/output for more clarification.

Sample Input

```
3
1
3
10
```

Sample Output

```
Case 1: 1
Case 2: 2
Case 3: 1073
```

13086 Pirates of the Mega Ocean

There are N island in the Mega Ocean. A_i is the number of people lived in island i. However, all islands are discrete from each other. There is no road transport system between the islands. Though they can travel each other by ship and boats, it is risky as the Pirates of the Mega Ocean create problems like kidnapping, demanding ransom, killing etc.

To get rid of this problem permanently, the governments of all these islands decided to make low cost two way tunnels between islands, so that they can visit each other without ship or boats and avoid getting in touch of Pirates of the Mega Ocean. However, the Pirates of the Mega Ocean got the information of tunnel building and sent a letter for all the Government chiefs demanding token money to let the governments building tunnels.

According to the letter, the governments have to pay $A_x * A_y$ amount of gold coins to build a tunnel between island x and island y. As the Pirates will not disturb anyone again if they get the money, the governments decided to pay the money. However, the governments of those islands want to spend as less as possible against the pirates. So they decided to build some tunnels in such a way that these tunnels connect all the islands (anyone can travel from any island to another through these tunnels) and the total money given to the pirates is as minimum as possible.

However, there is another problem. Some islands are so small that it is impossible to connect more than one tunnel with the island.

Given N, A and set of small island S, you have to find the minimum total gold coin you have to give to the pirates to build those tunnels, such that all the islands are connected and small islands are connected to at most one tunnel.

Input

First line of the input contains a positive integer T ($T \leq 200$), the number of test cases. First line of each test case contains two integer numbers N and M ($1 \leq N \leq 10^5$ and $0 \leq M < N$), denoting the number of island and the number of small islands respectively. Next line contains six integer numbers P, Q, R, X, Y and Z ($0 \leq P, Q, R, X, Y, Z \leq 10^5$). You have to calculate the number of people A_i for each island using the following equation:

$$A_i = (P \times i^2 + Q \times i + R)\%1000007, \quad 1 \leq i \leq N$$

Similarly, you have to calculate S, the set of small island as follows:

$$S_i = (X \times i^2 + Y \times i + Z)\%N + 1, \quad 1 \leq i \leq M$$

Note that, there can be duplicity in the small island set S.

Output

For each test case, print the test case number followed by the answer.

Explanation:

For both of the cases, $A_1 = 3$, $A_2 = 7$ and $A_3 = 13$.

For the first test case, there is no small island, so if you build tunnels A_1–A_2 and A_1–A_3, then the total cost will be $7 * 3 + 3 * 13 = 60$.

For the second test case, the only small island is $S_1 = 1$, so you can't build multiple tunnels with island 1. If you have to build tunnels A_1–A_2 and A_2–A_3, then the total cost will be $7 * 3 + 7 * 13 = 112$.

Sample Input

```
2
3 0
1 1 1 0 0 0
3 1
1 1 1 0 0 0
```

Sample Output

```
Case 1: 60
Case 2: 112
```

13087 Virus RNA

The whole world has become worried about the rapid spread of a virus. Scientists need to understand the folding of RNA of that virus so that they can have more information about its structure.

The basic RNA-folding problem is defined by a string S of length n over the four-letter alphabet {`A`, `U`, `C`, `G`}, and an integer d (distance parameter). Each letter in this alphabet represents an RNA nucleotide. Nucleotides `A` and `U` are called **complimentary** as are the nucleotides `C` and `G`. A matching consists of a set M of disjoint pairs of positions of S, i.e. in a set M no position i can be paired with two different positions j and j'. If pair (i, j) is in M, then the nucleotide at i-th position is said to match the nucleotide at position j. A match is a **permitted match** if the nucleotides at sites i and j are complimentary, $i < j$ and $|i-j| > d$. A matching M is non-crossing if and only if it does not contain any four sites $i < i' < j < j'$ where (i, j) and (i', j') are matches in M. Finally, a permitted matching M is a matching that is non-crossing, where each match in M is a permitted match. The basic RNA-folding problem is to find a permitted matching of maximum cardinality.

In this problem, you need to find the maximum cardinality of a permitted matching and the number of different sets M of that maximum cardinality. A set M is different from another set M' if there exists at least one pair (i, j) in M and (i', j') in M' such that either i and i' or j and j' are different.

Input

The first line of input file contains the number of test cases, T $(1 \le T \le 80)$. Then T cases follow:

Each case consists of two lines. The first line contains one integer: d $(0 \le d \le |S|)$. Then the second line contains the string S $(1 \le |S| \le 250)$. It will contain only the uppercase characters {`A`, `U`, `C`, `G`}.

Output

For each case, print '`Case` x`:` y z' in a separate line, where x is the case number, y is the maximum cardinality and z is the number of sets with maximum cardinality. As the value of z can be very large, print z *modulo* 10007.

Explanation of Sample cases:

For 1st case, there is no pair of positions which satisfies the conditions of permitted match, i.e. empty set is the only possible answer.

For 2nd case, the matches are shown below where the first position of a pair is denoted by '`(`' and the other position is denoted by '`)`':

```
GGACCUUUUGGACGC
((.((....)).).)
```

This is the only possible set with 4 permitted matches: {(1, 15), (2, 13), (4, 11), (5, 10)}.

Sample Input

```
2
1
AUA
4
GGACCUUUUGGACGC
```

Sample Output

```
Case 1: 0 1
Case 2: 4 1
```

13088 Lexicographically Smallest FPIS

A Permutation Insensitive String (**PIS**) is a string which does not change even if the positions of the characters are interchanged. For example, if the value of a **PIS** is '`abc`' it can also be written as '`acb`', '`bca`' etc. A Frequency Insensitive String (**FIS**) is a string whose value does not change if the frequency of any character is increased or decreased (Without altering the total length and without removing any character completely). So if the value of an **FIS** is '`aabc`' it can also be '`abbc`', or '`abcc`'. An **FPIS** (Frequency and Permutation Insensitive String) is a string that is both permutation and frequency insensitive. Given an **FPIS** you will have to write the lexicographically smallest version of it.

Input

First line of the input file contains an integer T ($T \leq 1000$) which denotes how many strings to follow. Each of the next T lines contains a single **FPIS** containing at most 1000 characters. All these characters are from lower case English alphabet.

Output

For each string in the input produce one line of output. This line contains lexicographically smallest version of the input **FPIS**.

The way you find lexicographic order is:

1. Compare the first (Leftmost) character of both strings. If they are different, order is given by the order of the two characters. ('`a`' is less than '`b`', '`y`' is (usually) less than '`z`')
2. If they are the same, move on to the next character of both strings.
3. If you run out of characters in one of the strings, that one comes first.
4. If you run out of characters in both strings, they are equal.

Sample Input

```
4
bca
pqab
aabb
c
```

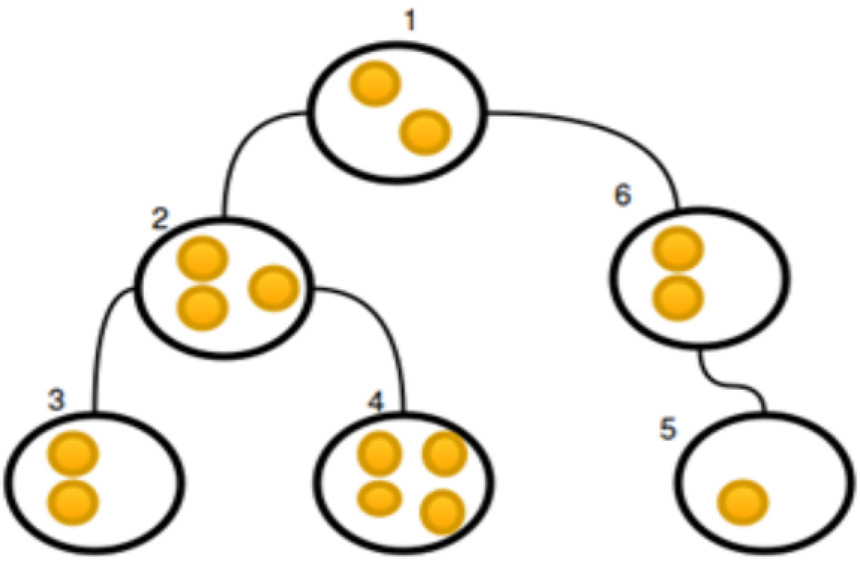
Sample Output

```
abc
abpq
aaab
c
```

13089 Golden Coins

Nik and Ann are playing a game. They have a tree with n nodes numbered from 1 to n.

Each node has 0 to m ($0 \leq m \leq 10^9$) golden coins.



Before the game starts they select a single node as the **treasure chest**. Suppose $d(u)$ denotes the distance between node u and the treasure chest.

The players make their move alternatively. In each move they do the followings:

- Select any single node u which has positive number of coins in it.
- Pick one coin from that node and move it to any node v such that $d(v) < d(u)$.

The game ends when no one can make a move (when all the coins are in the chest). Whoever can't make a move loses the game. Ann always makes the first move and both play optimally.

As Nik is a good programmer, he knows that if the treasure chest is chosen carefully, he can always win the game.

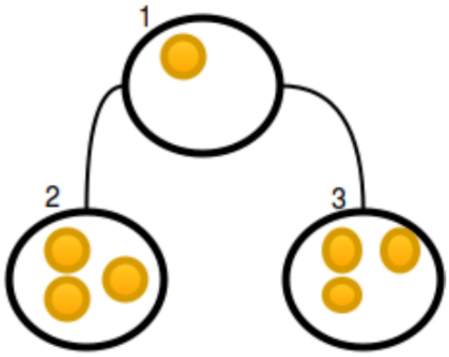
Find number of ways the treasure chest can be chosen so that Nik always wins the game.

Input

First line will contain an integer number T ($1 \leq T \leq 100$) denoting number of test cases. First line of each test case will consist of a single integer n ($1 \leq n \leq 1000$). Next line will contain n integers, where i-th integer denotes number of coins in node i. Each of the next $n-1$ lines will contain two integer numbers u and v denoting an edge of the tree.

Output

For each case, print case number and number of ways the treasure chest can be chosen so that Nik always wins the game.



Sample input looks like this:

Nik can win only if node **1** is chosen as treasure chest.

Sample Input

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

Sample Output

```
Case 1: 1
```

13090 Base of MJ

We know, if we want to check whether a decimal number is divisible by 3, we need to find the sum of digits of that number. If the sum is divisible by 3, then the original number will also be divisible by 3.

It took me a while to prove this. And then I realized this is true not only for 3 but for some other numbers as well. Sometimes not **only** for decimals but also for numbers in other bases as well. Can you find them?

In particular, given a particular divisor D, you will have to find how many valid different bases B, less or equal to $BMAX$, are possible such that when we represent any number N in base B and the sum of digits of N is S, the following implication is true:

N is divisible by D IF AND ONLY IF S is divisible by D.

For example, if $BMAX = 10$, $D = 3$, the answer is 3. The bases are 4, 7 and 10.

Input

First line will contain T ($T \leq 10000$), no of test cases. T lines will follow each with two integers $BMAX$ ($2 \leq BMAX \leq 10^{18}$) and D ($1 \leq D \leq 10^{18}$). You can assume that base of a number system is positive and not less than 2.

Output

For each case print one line, '`Case` C`:` A', where C is the case no and A is the required answer. Look at the output for sample input for details.

Sample Input

```
2
10 3
20 3
```

Sample Output

```
Case 1: 3
Case 2: 6
```

13091 No Ball

In the sport of cricket, a No Ball is a penalty against the fielding team, usually as a result of an illegal delivery by the bowler. The delivery of a No Ball gives an extra run to the opponent team. It might also give a warning under some regulations — even a suspension if it is extreme, and an additional ball must be bowled. In addition, the only way a batsman can get out in a No Ball is run out. In Twenty20s and recently one-day cricket matches, even in some amateur cricket, a batsman receives a 'free hit' on the ball after No Ball due to over stepping. This means the batsman can freely hit the ball with no danger of being out in certain ways. No Balls are not uncommon, especially in short form cricket, and fast bowlers tend to bowl them more often than spin bowlers, due to their longer run-up.

A No Ball may be called for many reasons. The most common No Ball is front foot No Ball. If the bowler bowls without some part of the front foot landing either grounded or in the air behind the popping crease or behind the line (as in the picture above).

However, umpires (on field judge in cricket) now-a-days, making mistakes to call a No Ball and teams are getting extra benefit for their mistakes. So ICPC (International Cricket Problem-solver Committee) wants to automate the process of calling No Ball. They will set a camera over the position of the line and take picture during bowling and detect the No Ball in real time.

ICPC wants to give you to write a program to detect the No Ball in real time. They will provide you the image data as a 5×5 grid as below,

```
..|..          ..|..
..|>.          .>|..
..|..          ..|..
..|..          ..|..
..|..          ..|..

No Ball      Not a No Ball
```

Here '`.`' is an empty field cell, '`|`' represents the line and '`>`' represents the bowler's front foot.

But if the camera is malfunctioning sometimes it gives 180 degree clockwise rotated images like below,

```
..|..          ..|..
..|>.          ..|..
..|..          ..|..
..|..          .<|..
..|..          ..|..

Normal      180 degree rotated
```

Here, '`|`' represents the line, '`>`' and '`<`' represents the front foot of bowler facing right and left respectively. According to the rules of No Ball and given image grid, we can say that, if the line is behind the front foot in any image then it is a No Ball, otherwise not.

Given an image as a 5×5 grid mentioned above, you have to detect whether the delivery is a No Ball or not.

Input

First line of the input contains an integer T ($T \leq 50$), number of test cases. Each test case consists of 5 lines describing the 5×5 grid. The grid will consist of '`.`' (dot), '`|`', '`<`' and '`>`'. There will be no other characters or symbols in the input. There will be exactly one of the characters from '`<`' and '`>`' in the input. There will be exactly one column with all '`|`', which represents the popping crease line.

There will be a blank line after each test case.

Output

For each test case, print the test case number and print '`No Ball`' if the corresponding grid represents a No Ball. Otherwise, print '`Thik Ball`'.

Sample Input

```
3
..|..
..|>.
..|..
..|..
..|..

..|..
.>|..
..|..
..|..
..|..

.|...
.|...
.|...
.|.>.
.|...
```

Sample Output

```
Case 1: No Ball
Case 2: Thik Ball
Case 3: No Ball
```

UVa Online Judge

13092 Fold the String

Alice took File Compression as her term project. Supervisor suggested her to do everything on her own. She devised her own algorithm for compression. Her algorithm converts any file to its string representation S. Then she encodes the string with series of operations. She starts processing the string from its end. At any step she does any of the following two operations:

1. Encode the last character of the remaining non-encoded string. This last character encoding needs x unit of memory.

   Encode$(S[1, 2, 3, \ldots, n])$ = **Encode**$(S[1, 2, 3, \ldots, n-1])$ + **information of** $S[n]$.

   Where n is the length of string S.

2. Encode a suffix of the remaining non-encoded string. She can encode any suffix that satisfies the folding property. Suffix starting at index i has the folding property if it's the mirror of a substring ending at index $i-1$. For example, both **suffix**("`aabbaabba`", 8) and **suffix**("`aabbaabba`", 6) have the folding property. Any possible suffix encoding needs y unit of memory.

   Here, **suffix**(S, i) is the suffix of string S starting at index i.

   Encode$(S[1, 2, 3, \ldots, i-1, i, \ldots, n-1, n])$ = **Encode**$(S[1, 2, 3, \ldots, i-1])$ + **information of suffix**(S, i).

   Where, **suffix**(S, i) satisfies the folding property.

Alice wants to know the performance of her compression algorithm. She has collection of files for performance testing. She converted each file to its corresponding string representation. Now, needs your help in determining the total memory unit each file needs after compression.

Input

Input starts with a line with number of test cases T $(1 \leq T \leq 25)$. Each of the following T lines has information about a single file. Each line has two integer x, y $(1 \leq x, y \leq 1000)$ and a string S $(1 \leq |S| \leq 1000000)$. S comprises only of lowercase letter.

Output

Output contains T lines. Each of them is in format '`Case` t`:` c'. t is the test case number and c is the amount of memory needed by the compressed string.

Test Case Analysis:

Optimal compression steps of case 1 are "`aabbaa`" `->` "`aab`", "`aab`" `->` "`aa`", "`aa`" `->` "`a`", "`a`" `->` "". First and third step require cost 1 because of suffix encoding. Second and fourth step require cost 2. So, the total cost is $1 + 2 + 1 + 2 = 6$.

Sample Input

```
2
2 1
aabbaa
1 1
aabbaabba
```

Sample Output

```
Case 1: 6
Case 2: 5
```