2501. Score

Time Limit: 1.0 Seconds   Memory Limit: 65536K
Total Runs: Accepted Runs:

There is an objective test result such as "OOXXOXXOOO". An 'O' means a correct answer of a problem and an 'X' means a wrong answer. The score of each problem of this test is calculated by itself and its just previous consecutive 'O's only when the answer is correct. For example, the score of the 10th problem is 3 that is obtained by itself and its two previous consecutive 'O's.

Therefore, the score of "OOXXOXXOOO" is 10 which is calculated by "1+2+0+0+1+0+0+1+2+3".

You are to write a program calculating the scores of test results.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing a string composed by 'O' and 'X' and the length of the string is more than 0 and less than 80. There is no spaces between 'O' and 'X'.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line is to contain the score of the test case.

The following shows sample input and output for five test cases.

Sample Input

```
5
OOXXOXXOOO
OOXXOOXXOO
OXOXOXOXOXOXOX
OOOOOOOOOO
OOOOXOOOOXOOOOX
```

Output for the Sample Input

```
10
9
7
55
30
```

Source: South Korea 2005 - Seoul

2502. Digit Generator

Time Limit: 1.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

For a positive integer N, the digit-sum of N is defined as the sum of N itself and its digits. When M is the digitsum of N, we call N a *generator* of M.

For example, the digit-sum of 245 is 256 (= 245 + 2 + 4 + 5). Therefore, 245 is a generator of 256.

Not surprisingly, some numbers do not have any generators and some numbers have more than one generator. For example, the generators of 216 are 198 and 207.

You are to write a program to find the smallest generator of the given integer.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case takes one line containing an integer N, $1 \le N \le 100{,}000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line is to contain a generator of N for each test case. If N has multiple generators, print the smallest. If N does not have any generators, print 0.

The following shows sample input and output for three test cases.

Sample Input

```
3
216
121
2005
```

Output for the Sample Input

```
198
0
1979
```

Source: South Korea 2005 - Seoul

2503. Apple Tree

Time Limit: 1.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

Figure 1 shows an apple tree in which each node has exactly one apple and every internal node has at least two children nodes.

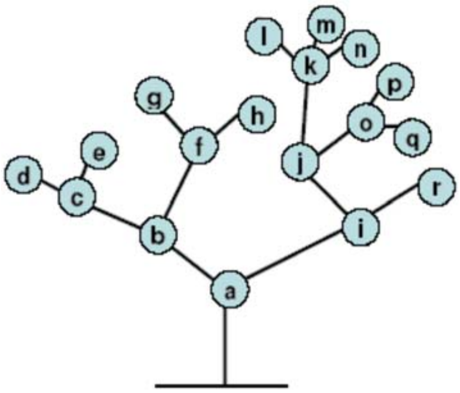
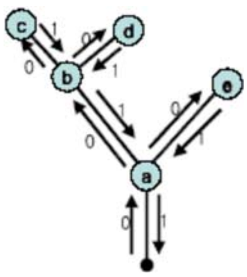


Figure 1

Consider a worm in the apple tree. The worm visits apples (i.e., nodes) following the order of depth-first traversal from the root of the apple tree. (Notice that, the branching priority of an internal node is left-first order.) Figure 2 shows a binary representation of the depth-first traversal by a worm in the apple tree.



Binary representation	0	0	0	1	0	1	1	0	1	1
first visited node	a	b	c		d			e		
returned node				c		d	b		e	a

Figure 2

The binary representation is defined as follows: During the depth-first traversal,

"0" is printed, if a new node is visited, and
"1" is printed, if a node is returned after visiting all descendent nodes.

The second line of the table shows the first visited node corresponding to each "0" and the third line shows the returned node corresponding to each "1". For example, the apple tree in Figure 1 can be represented by the definition above.

Binary representation	0	0	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	1	1	0	1	1	1
first visited node	a	b	c	d		e			f	g		h				i	j	k	l		m		n			o	p		q				r			
returned node					d		e	c			g		h	f	b					l		m		n	k			p		q	o	j		r	i	a

If there are wormy apples in the apple tree, we should remove them. When we cut a node z, every apple in the subtree rooted at node z (including the apple of node z) is going down. For example, apples *k*, *l*, *m*, and *n* are going down by cutting node *k* and apples *b*, *c*, *d*, *e*, *f*, *g* and *h* are going down by cutting node *b* in the apple tree in Figure 1. In order to remove wormy apples, we should find the subtree rooted at a node in which every wormy apple exists, and then cut the node.

Given two wormy apples (possibly same) in an apple tree, the problem is to find a node satisfying that two wormy apples must drop and the minimum number of normal apples drop simultaneously by cutting the node. For example, suppose that nodes *c* and *f* are given as wormy apples in the apple tree in Figure 1. By cutting nodes *a* or *b* , two wormy apples can be removed. However, the smaller number of normal apples drop by cutting node *b* than the case that we cut node *a*. So, we want to find the node *b*.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case is composed of three lines. In first line, the number of nodes N ($3 \le N \le 2000$) is given. In next second line, $2N$ binary representation of an apple tree is given. There is no space between each representation. In last third line, two indices A and B ($1 \le A,B \le 2N$) corresponding to two wormy apples in the binary representation are given. These two indices are not corresponding to node directly but index of binary representation. Notice that, the value of each index can be "0" or "1".

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, find a node satisfying that the wormy apples must be removed and the minimum number of normal apples are removed simultaneously by cutting the node, and then print an index of "0" where the node was first visited and the other index of "1" where the node was returned in the binary representation. Print exactly one line for each test case index of "0" first, and index of "1" later. And there must be a single space between two indices.

The following shows sample input and output for three test cases.

Sample Input

```
3
18
000010110010111000010101100101110111
4 11
18
000010110010111000010101100101110111
11 12
18
000010110010111000010101100101110111
5 12
```

Output for the Sample Input

```
2 15
9 14
2 15
```

Source: South Korea 2005 - Seoul

2504. Pinary

Time Limit: 1.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

"Pinary" number is a positive number using only two digits "0" and "1" with usual rule that it must not begin with a 0, and the additional rule that two successive digits must not be both "1". This means that the factor "11" is forbidden in the string. So the allowed Pinary writings are 1, 10, 100, 101, 1000, 1001, ..., 100010101010100010001. For example, "100101000" is a Pinary number, but neither "0010101" nor "10110001" are Pinary numbers.

Each Pinary number represents a positive integer in the order they appear (using length order, lexicographic order), that is, 1 is mapped to 1, 10 is mapped to 2. And 100, 101 and 1000 are mapped to 3, 4 and 5, respectively. You are to write a program to generate Pinary number representations for integers given.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing a postive integer $2 < K < 90,000,000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print the Pinary number representation for input integer.

The following shows sample input and output for three test cases.

Sample Input

```
3
7
2000
22
```

Output for the Sample Input

```
1010
1001000001001000
1000001
```

Source: South Korea 2005 - Seoul

2505. PushPush

Time Limit: 2.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

A treasure hunter got a map of an ancient pyramid where the sacred treasure is enshrined. He started on a journey to excavate the treasure. Once he arrived at the gate of the pyramid, the earthquake occurred. As a result, some rocks fell from the ceiling to the floor on the inner pyramid. He found two rocks blocked the way to the treasure. He couldn't pull any rocks because they are too large and heavy. One thing he could do is to push a single rock moving one block forward. Now, he is going to reach to the treasure and pushing the rocks if necessary. If he will push a rock to a wrong direction, he cannot advance any more, thus he will have to give up the treasure.

In a map of Figure 1, white blocks represent open paths and black blocks represent barriers. Black circles represent rocks, and *E* and *T* represent the position of the entrance and the treasure, respectively. A person and rocks can move to a direction of north, east, west, and south. In order to push a rock, the forward and the backward blocks of the position of the rock should be both white. In Figure 1, the person on *E* can push the rock on (2, 3) to north or east. But, he cannot push it to west and south because he can't move to the positions (2, 4) and (1, 3). If he will push it to east, he cannot push it to east any more because he cannot push two rocks at the same time. If he will push the rock on (2, 5) to north, the treasure will be broken down. In order to get the treasure, he has to push the rock on (2, 3) to north and then push the rock on (2, 5) to east. Notice that any rock cannot be moved outside the pyramid.

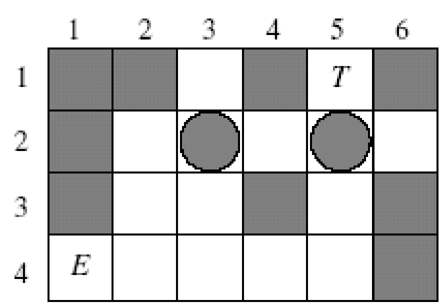


Figure 1

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

Figure 2

A map is represented by a matrix as Figure 2. In a matrix, 0, 1, 2, 3, and 4 represent an open path, a barrier, the entrance, the treasure, and a rock, respectively. The entrance, the treasure, and the rocks are positioned on open paths in the map.

Write a program for computing the minimum number of pushing rocks so as to reach to the treasure. For an example in Figure 1, the answer is 2. If there is a path to the treasure without pushing any rocks, the answer is 0. If there doesn't exist a path, the answer is -1.

Input

Your program is to read from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. Each test case starts with a line containing two integers *n* and *m*, the number of rows and the number of columns of a map, $2 \leq n, m \leq 50$. Each of the following *n* lines contains *m* integers of 0, 1, 2, 3, or 4, they represent each row of the map, where 0, 1, 2, 3, and 4 represent an open path, a barrier, the entrance, the treasure, and a rock, respectively. There are exactly a single 2, a single 3, and two 4's in the map. There is a single space between the integers.

Output

Your program is to write to standard output. Print exactly one line for each test case. Print the minimum number of pushing rocks if they can get to the treasure. Otherwise, print `-1`.

The following shows sample input and output for three test cases.

Sample Input

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

Output for the Sample Input

```
2
-1
0
```

Source: South Korea 2005 - Seoul

2506. Seminar Room

Time Limit: 2.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

My department, Department of Computer Science, has a seminar room equipped with luxurious conveniences. Many research groups want to utilize the seminar room among others. To make out a weekly schedule of the seminar room, each research group was requested to submit a candidate time interval for its own seminar. Having realized that some candidate time intervals were overlapped, the department chairman Prof. Yang fell in a difficult situation. In order to assign the seminar room to the groups as fair as possible, he decided to request the seminar groups to submit two candidate time intervals. The two time intervals submitted by a group can be overlapped; however, one should not be contained in the other. Now, Prof. Yang's problem of preparing a weekly schedule is to assign one of the candidate time intervals submitted by each group to the group so that the time intervals assigned to groups do not overlap each other. If the finish time of one time interval coincides with the start time of the other, the two time intervals are not considered to be overlapped.

A candidate time interval is specified by a pair of the start time and finish time. The time is written in format *ddd*:*hh*:*mm* , where *ddd* are three capital letters representing the day of a week, *hh* are two digits representing full hours ($00 \le hh \le 23$), and *mm* are two digits representing minutes ($00 \le mm \le 59$). On Sundays, a reservation will not be made. Thus, *ddd* are one of MON, TUE, WED, THU, FRI, SAT.

For example, suppose you are given three seminar groups and its candidate time intervals as shown in the table below.

seminar group 1	MON:09:00 MON:11:00
	MON:10:00 MON:12:00
seminar group 2	MON:09:00 MON:11:30
	TUE:13:25 TUE:14:27
seminar group 3	MON:09:30 MON:11:00
	MON:23:00 TUE:01:00

If three time intervals [MON 09:00 - MON 11:00], [TUE 13:25 - TUE 14:27], and [MON 23:00 - TUE 01:00] are assigned to seminar groups 1, 2, and 3, respectively, the time intervals do not overlap each other, and thus all the seminar groups will be happy.

Now, Prof. Yang wants to find an efficient method to determine, for given two candidate time intervals per each seminar group, whether or not one of the candidate time intervals can be assigned to each group so that all the time intervals assigned to groups do not overlap each other. Write a program that can help Prof. Yang.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing a positive integer n , $n \leq 1,000$, representing the number of seminar groups. In the next n lines, each line contains two candidate time intervals for a seminar group in the form $d_1d_1d_1:h_1h_1:m_1m_1$, $d_2d_2d_2:h_2h_2:m_2m_2$, $d_3d_3d_3:h_3h_3:m_3m_3$, and $d_4d_4d_4:h_4h_4:m_4m_4$. The two candidate time intervals are [$d_1d_1d_1:h_1h_1:m_1m_1$ - $d_2d_2d_2:h_2h_2:m_2m_2$] and [$d_3d_3d_3:h_3h_3:m_3m_3$ - $d_4d_4d_4:h_4h_4:m_4m_4$]. There is a single space between the start time and finish time of a time interval and between the two time intervals.

Output

Your program is to write to standard output. Print exactly one line for each test case. Print `YES` if one of the candidate time intervals can be assigned to each group so that all the time intervals assigned to groups do not overlap each other. Otherwise, print `NO`.

The following shows sample input and output for three test cases.

Sample Input

```
3
3
MON:09:00 MON:11:00 MON:10:00 MON:12:00
MON:09:00 MON:11:30 TUE:13:25 TUE:14:27
MON:09:30 MON:11:00 MON:23:00 TUE:01:00
2
SAT:08:00 SAT:09:00 FRI:13:13 FRI:14:14
SAT:08:00 SAT:09:00 FRI:13:13 FRI:14:14
4
FRI:13:13 FRI:14:14 SAT:08:00 SAT:09:00
FRI:13:13 FRI:14:14 SAT:08:00 SAT:09:00
FRI:13:13 FRI:14:14 SAT:08:00 SAT:09:00
FRI:13:13 FRI:14:14 SAT:08:00 SAT:09:00
```

Output for the Sample Input

```
YES
YES
NO
```

Source: South Korea 2005 - Seoul

2507. Cannon's Move

Time Limit: 2.0 Seconds   Memory Limit: 65536K
Total Runs: Accepted Runs:

A game called *janggi* is a variant of chess played in Korea. It is played by two players on a board with 9 × 10 size. Each player has 16 pieces of 7 kinds — 1 king, 2 chariots, 2 cannons, 2 horses, 2 elephants, 2 guards and 5 pawns. In janggi, the cannon's move has several restrictions. The cannon moves and captures the opponent's piece by jumping over exactly one piece along a straight line.

Here you are to write a program for the one-dimensional janggi. The board consists of linear cells. Four kinds of pieces are given: C (cannon), E (enemy), F (friend) and K (king). C and F are your pieces, and E and K are the opponent's ones. Only one C and one K are given. Your have to capture K using a sequence of valid moves.

The rules of one-dimensional janggi are:

1. You can move only C.
2. C can move by jumping over exactly one piece of K, E or F.
3. The next position of C is either an unoccupied position or a position occupied by opponent's pieces.
4. When C moves to a position occupied by an opponent's piece, we say "C captures E (or K)" and the captured piece is removed from the board. Obviously C can not capture a friendly piece, F.
5. The game is over when C captures K.

For example, in the following board, C is in position 6. In this position, C can move to 2 (by capturing E), 3, 4, 9 or 10 (by capturing E). Other positions are not reachable by a single move.

	E			F	C		E		E	F		K	
1	2	3	4	5	6	7	8	9	10	11	12	13	14

After C captures E in 10, the board changes as follows. Now C can move to 6, 7, 12 or 13 (by capturing K).

	E			F			E		C	F		K	
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case takes one line containing a string which represents the configuration of the one-dimensional janggi board. Occupied cells are denoted by C, E, F and K; and unoccupied cells are denoted by B. The length of the string is at least 5 and at most 200.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print the minimum number of moves to capture K. If it is impossible to capture K, print 0. The following shows sample input and output for three test cases.

Sample Input

```
3
BEBBFCBEBEFBKB
BEEBKCE
BCBEFFEK
```

Output for the Sample Input

```
2
3
0
```

Source: South Korea 2005 - Seoul

2508. Bond

Time Limit: 2.0 Seconds Memory Limit: 65536K Special Judge
Total Runs: Accepted Runs:

Tom is a chemist who has an interest in molecular structure. Recently he found a new atom and named it *tomatom*. Two or more tomatoms are joined by bonds to make a molecule called *tomolecule*. In chemistry, the term *valency* is used to indicate the number of bonds joining an atom to its neighboring atoms. Tomatoms in a tomolecule are joined satisfying the following conditions:

1. Each tomatom is joined by at least one bond, and there can be multiple bonds between two tomatoms.
2. A tomolecule is connected, that is, does not have isolated components. That means there is a path composed of bonds between every two tomatoms in a tomolecule.
3. There is no bond between a tomatom and itself.

Currently Tom is trying to investigate the property of valency in a tomolecule. Precisely speaking, given some positive integers, Tom wants to know whether there can be a tomolecule whose valency numbers of tomatoms are same to the integers and satisfying the above conditions.

For example, suppose a tomolecule is composed of three tomatoms and given integers are 50, 60, and 70. For convenience, we name three tomatoms as A^1 , A^2 , A^3 . If there are 20 bonds between A^1 and A^2 , 30 bonds between A^1 and A^3 , and 40 bonds between A^2 and A^3 , the valency of A^1 , A^2 , A^3 are 50, 60, 70, respectively. So, there can be a tomolecule corresponding to such integers. If 10, 20, 40 are given, it is easy to know that there can be no tomolecule whose valency numbers of tomatoms are such integers. As another example, suppose four tomatoms and four integers 1, 1, 1, 1 are given, there can be no tomolecule satisfying the above condition (2).

You are to write a program to help Tom. For given positive integers, your program should determine whether there can be a tomolecule whose valency numbers of tomatoms are same to the integers and satisfying the above conditions. Also, in case such a tomolecule exists, your program should find one possible structure.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer N , the number of tomatoms, $2 \le N \le 200$. In the next line, N nondecreasing positive integers are given. Assume k-th integer in the line represents the valency of tomatom A^k , $1 \le k \le N$. There is a single space between the integers, and the integers are between 1 and 10,000, both inclusive.

Output

Your program is to write to standard output. For each test case, print YES at the first line if there can be a tomolecule corresponding to the integers in the input. Otherwise, print NO. If your program printed YES at the first line, your program should find one possible structure and print the number of bonds between tomatom A^k and tomatoms A^{k+1} , A^{k+2} , ... , A^N at $(k+1)$-th line, $1 \le k \le N-1$. There should be a single space between the numbers.

The following shows sample input and output for five test cases.

Sample Input

```
5
3
50 60 70
3
10 20 40
4
1 1 1 1
6
1 1 1 1 1 5
6
1 1 1 1 4 4
```

Output for the Sample Input

```
YES
20 30
40
NO
NO
YES
0 0 0 0 1
0 0 0 1
0 0 1
0 1
1
YES
0 0 0 1 0
0 0 1 0
0 0 1
0 1
2
```

Note: Special judge problem, you may get "Wrong Answer" when output in wrong format.

Source: South Korea 2005 - Seoul

2509. Roof

Time Limit: 3.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

You are going to build a new house. To know the minimum construction cost of the roof, you want to know its height. But you have the information only on the boundary of the house, so you have to compute the height of the roof from the boundary information. The formal description is as follows. The boundary of a house is defined as a rectilinear polygon with only axis-parallel edges, horizontal or vertical. Let P be the rectilinear polygon with n vertices. A *straight skeleton* SK(P) is a trace of the vertices of P when P is shrunk, each edge moving at the same speed and keeping the same direction. Figure 1 (a) and (b) show the shrinking process for P and SK(P), respectively.

Now we assume that P is on the XY plane, and P is shrunk at unit speed while moving upward (+Z-direction) at unit speed. Then P traces a three dimensional polyhedral surface, called *terrain*, and SK(P) can be seen as the projection of the edges of this terrain onto the XY plane. From the definition, each face of this terrain lies inside a plane that makes a dihedral angle 45° with XY plane. Another fact for this terrain is that each of its faces is bounded by at least one edge of P . We call this terrain the *roof* of the polygon P . For instance, Figure 1 (c) shows the roof. The height of a point q on the roof is the altitude, i.e., distance between q and its projected point on the XY plane. The height of a roof is the maximum height of points on the roof. Your task is to compute the height of the roof of the input rectilinear polygon P .

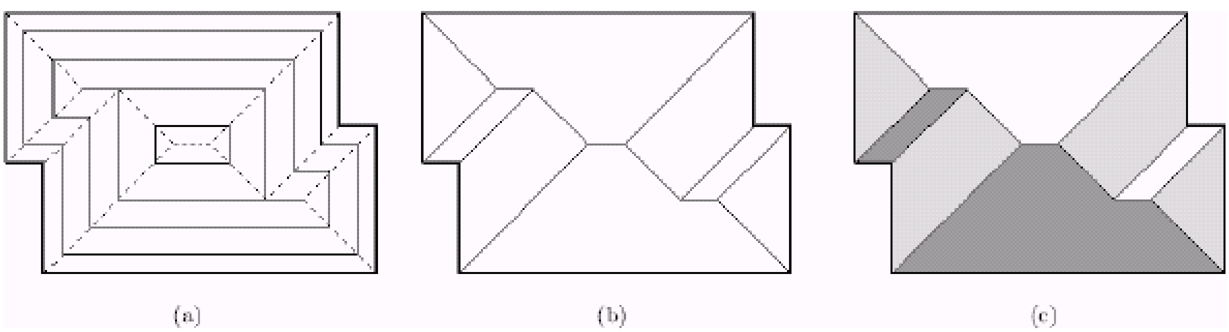


Figure 1

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer n , the number of vertices of the input polygon P , $4 \le n \le 1{,}000$. In the next lines, (x, y)-coordinates of the n vertices of P are given according to the counterclockwise order. The first pair of integers is x and y coordinates of the first vertex of P , and the second pair is x and y coordinates of the second vertex of P , and so on. The coordinates are separated by a single space, and are positive integer values between 1 and 100,000, both inclusive.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print the height of the roof of P with exactly one digit in the fraction part.

The following shows sample input and output for three test cases.

Sample Input

```
3
4
1 1 4 1 4 4 1 4
6
4 1 8 1 8 6 1 6 1 4 4 4
14
1 1 4 1 4 2 7 2 7 1 11 1 11 10 8 10
8 8 1 8 1 6 3 6 3 4 1 4
```

Output for the Sample Input

```
1.5
2.0
3.0
```

Source: South Korea 2005 - Seoul

2510. String Compression

Time Limit: 2.0 Seconds Memory Limit: 65536K
Total Runs: Accepted Runs:

Run Length Encoding(RLE) is a simple form of compression. RLE consists of the process for searching for a repeated runs of a single character in a string to be compressed, and replacing them by a single instance of the character and a run count. For example, a string `abcccddddddefgggggggggghijk` is encoded into a string `ab3c6def10ghijk` by RLE.

A new compression method similar to RLE is devised and the rule of the method is as follows: if a substring *S* is repeated *k* times, replace *k* copies of *S* by *k*(*S*). For example, `letsgogogo` is compressed into `lets3(go)`. The length of `letsgogogo` is 10, and the length of `lets3(go)` is 9. In general, the length of *k*(*S*) is (number of digits in *k*) + (length of *S*) + 2 (for '(' and ')'). For example, the length of `123(abc)` is 8. It is also possible to nest compression, so the substring *S* may itself be a compressed string. For example, `nowletsgogogoletsgogogo` could be compressed as a `now2(lets3(go))`, and `nowletsgogogoletsgogogoandrunrunrun` could be compressed as `now2(lets3(go))and3(run)`.

Write a program that, for a given string, gives a shortest compressed string using the compression rules as described above.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of a single line containing one string of no more than 200 characters drawn from a lower case alphabet. The length of shortest input string is 1.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print the length of the shortest compressed string.

The following shows sample input and output for four test cases.

Sample Input

```
4
ababcd
letsgogogo
nowletsgogogoletsgogogo
nowletsgogogoletsgogogoandrunrunrun
```

Output for the Sample Input

```
6
```

```
9
15
24
```

Source: South Korea 2005 - Seoul
