UVa Online Judge

10337 Flight Planner

Calculating the minimal cost for a flight involves calculating an optimal flight-altitude depending on wind-strengths changing with different altitudes. It's not enough just to ask for the route with optimal wind-strength, because due to the mass of a plane you need a certain amount of fuel to rise. Moreover due to safety regulations it's forbidden to fly above a certain altitude and you can't fly under zero-level.

In order to simplify the problem for now, we assume that for each 100 miles of flight you have only three possiblities: to climb one mile, to hold your altitude or to sink one mile.

Climb flight requires 60 units of fuel, holding your altitude requires 30 units and sinking requires 20 units.

In the case of headwind you need more fuel while you can save fuel flying with tailwind. Windstrength w will satisfy the condition $-10 \leq w \leq 10$, where negative windstrength is meant to be headwind and positive windstrength is tailwind.

For one unit of tailwind you can save one unit of fuel each 100 miles; each unit of headwind will cost an extra unit of fuel.

For example to climb under conditions of windstrength $w = -5$, you need 65 units of fuel for this 100 miles.

Given the windstrengths on different altitudes for a way from here to X, calculate the minimal amount of fuel you need to fly to X.

Input

The first line of the input file contains the number N of test cases in the file. The first line of each test case contains a single integer X, the distance to fly, with $1 \leq X \leq 100000$ miles and X is a multiple of 100. Notice that it's not allowed to fly higher than 9 miles over zero and that you have to decide whether to climb, hold your altitude or to sink only for every 100 miles.

For every mile of allowed altitude (starting at altitude 9 down to altitude 0) there follow $\frac{X}{100}$ windstrengths, starting with the windstrength at your current position up to the windstrength at position $X - 100$ in steps of 100 miles. Test cases are separated by one or more blank lines.

Output

For each test case output the minimal amount of fuel used flying from your current position (at altitude 0) to X (also at altitude 0), followed by a blank line.

Sample Input

```
2

400
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

```
1 9 9 1
1 -9 -9 1

1000
 9  9  9  9  9  9  9  9  9  9
 9  9  9  9  9  9  9  9  9  9
 9  9  9  9  9  9  9  9  9  9
 9  9  9  9  9  9  9  9  9  9
 9  9  9  9  9  9  9  9  9  9
 9  9  9  9  9  9  9  9  9  9
 7  7  7  7  7  7  7  7  7  7
-5 -5 -5 -5 -5 -5 -5 -5 -5 -5
-7 -3 -7 -7 -7 -7 -7 -7 -7 -7
-9 -9 -9 -9 -9 -9 -9 -9 -9 -9
```

Sample Output

```
120

354
```

UVa Online Judge

11022 String Factoring

Spotting patterns in seemingly random strings is a problem with many applications. E.g. in our efforts to understand the genome we investigate the structure of DNA strings. In data compression we are interested in finding repetitions, so the data can be represented more efficiently with pointers. Another plausible example arises from the area of artificial intelligence, as how to interpret information given to you in a language you do not know. The natural thing to do in order to decode the information message would be to look for recurrences in it. So if the SETI project (the Search for Extra Terrestrial Intelligence) ever get a signal in the H21-spectra, we need to know how to decompose it.

One way of capturing the redundancy of a string is to find its factoring. If two or more identical substrings A follow each other in a string S, we can represent this part of S as the substring A, embraced by parentheses, raised to the power of the number of its recurrences. E.g. the string DOODOO can be factored as $(DOO)^2$, but also as $(D(O)^2)^2$. Naturally, the latter factoring is considered better since it cannot be factored any further. We say that a factoring is irreducible if it does not contain any consecutive repetition of a substring. A string may have several irreducible factorings, as seen by the example string POPPOP. It can be factored as $(POP)^2$, as well as $PO(P)^2OP$. The first factoring has a shorter representation and motivates the following definition. The weight of a factoring, equals the number of characters in it, excluding the parentheses and the exponents. Thus the weight of $(POP)^2$ is 3, whereas $PO(P)^2OP$ has weight 5. A maximal factoring is a factoring with the smallest possible weight. It should be clear that a maximal factoring is always an irreducible one, but there may still be several maximal factorings. E.g. the string ABABA has two maximal factorings $(AB)^2A$ and $A(BA)^2$.

Input

The input consists of several rows. The rows each hold one string of at least one, but less than 80 characters from the capital alphabet `A-Z`. The input is terminated by a row containing the character '`*`' only. There will be no white space characters in the input.

Output

For each string in the input, output one line containing the weight of a maximal factoring of the string.

Sample Input

```
PRATTATTATTIC
GGGGGGGGG
PRIME
BABBABABBABBA
ARPARPARPARPAR
*
```

Sample Output

```
6
1
5
6
5
```

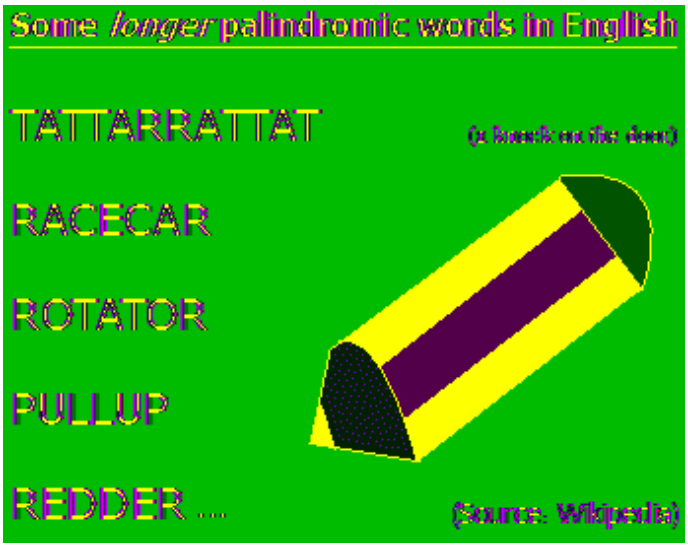
UVa Online Judge

11151 Longest Palindrome

A **palindrome** is a string that reads the same from the left as it does from the right. For example, I, GAG and MADAM are palindromes, but ADAM is not. Here, we consider also the *empty string* as a palindrome.

From any non-palindromic string, you can always take away some letters, and get a palindromic subsequence. For example, given the string ADAM, you remove the letter M and get a palindrome ADA.

Write a program to determine the length of the longest palindrome you can get from a string.



Input

The first line of input contains an integer T (≤ 60). Each of the next T lines is a string, whose length is always less than 1000.

For $\geq 90\%$ of the test cases, string length ≤ 255.

Output

For each input string, your program should print the length of the longest palindrome you can get by removing zero or more characters from it.

Sample Input

```
2
ADAM
MADAM
```

Sample Output

```
3
5
```

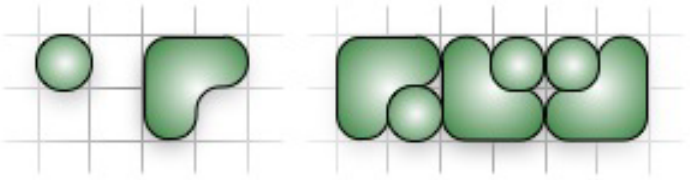
UVa Online Judge

11310 Delivery Debacle

Wolfgang Puck has two very peculiar habits:

- I. He only makes two shapes of cakes. One is square and has an area of one unit. The other is L-shaped and has an area of three units.
- II. He will only deliver cakes packed in very specific box sizes. The boxes are always 2 units wide and are of varying length.

One day Wolfgang wondered in how many different ways he can pack his cakes into certain sized boxes. Can you help him?



The precise sizes of the cakes Wolfgang makes and one way to pack them in a box of length 6.

The five ways to pack a box of length 2.

Input

The input begins with t, the number of different box lengths. The following t lines contain an integer n ($1 \leq n \leq 40$).

Output

For each n output on a separate line the number of different ways to pack a 2-by-n box with cakes described above. Output is guaranteed to be less than 10^{18}.

Sample Input

```
2
1
2
```

Sample Output

```
1
5
```

UVa Online Judge

11407 Squares

For any positive integer N, $N = a_1^2 + a_2^2 + \ldots + a_n^2$ that is, any positive integer can be represented as sum of squares of other numbers.

Your task is to print the smallest 'n' such that $N = a_1^2 + a_2^2 + \ldots + a_n^2$.

Input

The first line of the input will contain an integer 't' which indicates the number of test cases to follow.

Each test case will contain a single integer 'N' ($1 \leq N \leq 10000$) on a line by itself.

Output

Print an integer which represents the smallest 'n' such that $N = a_1^2 + a_2^2 + \ldots + a_n^2$.

Explanation for sample test cases:

5 $\rightarrow$ number of test cases
1 $= 1^2$ (1 term)
2 $= 1^2 + 1^2$ (2 terms)
3 $= 1^2 + 1^2 + 1^2$ (3 terms)
4 $= 2^2$ (1 term)
50 $= 5^2 + 5^2$ (2 terms)

Sample Input

```
5
1
2
3
4
50
```

Sample Output

```
1
2
3
1
2
```

UVa Online Judge

11813 Shopping

You have just moved into a new apartment and have a long list of items you need to buy. Unfortunately, to buy this many items requires going to many different stores. You would like to minimize the amount of driving necessary to buy all the items you need.

Your city is organized as a set of intersections connected by roads. Your house and every store is located at some intersection. Your task is to find the shortest route that begins at your house, visits all the stores that you need to shop at, and returns to your house.

Input

The first line of input contains a single integer, the number of test cases to follow. Each test case begins with a line containing two integers N and M, the number of intersections and roads in the city, respectively. Each of these integers is between 1 and 100000, inclusive. The intersections are numbered from 0 to $N-1$. Your house is at the intersection numbered 0. M lines follow, each containing three integers X, Y, and D, indicating that the intersections X and Y are connected by a bidirectional road of length D. The following line contains a single integer S, the number of stores you need to visit, which is between 1 and ten, inclusive. The subsequent S lines each contain one integer indicating the intersection at which each store is located. It is possible to reach all of the stores from your house.

Output

For each test case, output a line containing a single integer, the length of the shortest possible shopping trip from your house, visiting all the stores, and returning to your house.

Sample Input

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

Sample Output

```
4
```