Problem A: Rabbit

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

A rabbit hops along a road. For convenience we interpret the road as a sequence of squares that are numerated by integers $1, ..., N$. Some squares are blocked by wolves, so the poor rabbit has to be very careful to avoid them. Also there are squares that have grass and carrots growing on them, so on these the rabbit gets some rest and feed.

The rabbit starts his journey from square 1 and wants to get on square N, of course safe and sound without meeting wolves and at the same time visiting as many squares with grass and carrots as possible. The rabbit has some limited features, from square k it can only jump on squares $k+1$, $k+3$ and $k+5$. You are to find out how many squares with grass and carrots the rabbit is able to reach on his journey.

Input

The input consists of

- one line containing N ($2 \leq N \leq 1000$) – the number of squares
- one line containing N characters, with the i-th character corresponding to square i. Character 'w' represents fangs of wolves, '"' the grass and '.' an ordinary empty square. The first and last square are guaranteed to be empty and the rabbit is required to start in square 1 and end up in square N.

Output

Output just one integer - the maximal number of squares with grass that the rabbit is able to visit. If it's impossible to reach square n output -1.

Sample input and output

Input	Output
4 ."".	2
5 .w"..	0
9 .www.www.	-1

Problem B: Reading

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

There's a huge blizzard raging outside. It is becoming so bad that a few minutes ago the electricity in your house went out, so you won't be able to use your computer, probably for the rest of the day. Instead you decide to read some of your old comic books that are still neatly lined up on a bookshelf above your desk.

While staring thoughtfully at the shelf, you remember that back in the days (before you owned a computer) you rated all of your comic books for how reading them affects your mood and wrote the results on the backs of the books. Because your mood dropped to -100 due to the power failure, you want to read books that increase your mood as much as possible.

Unfortunately your OCD forces you to follow some very specific rules when picking books to read: You go through them from left to right, starting by reading the leftmost one. While your mood is still negative, you may only skip one book at a time (i.e. after reading the i-th one you may only chose whether to continue with the $(i + 1)$-th or the $(i + 2)$-th). When it has increased to a nonnegative number you are also able to skip two books at a time, at least as long as your mood stays nonnegative (it might fall below zero again when you are forced to read a comic that decreases your mood).

Before you start reading you want to know to what level you can increase your mood by starting with the leftmost book and following the rules until you have reached (and read!) the rightmost one.

Input

The input consists of

- one line containing n ($3 \leq n \leq 10^5$) – the number of comic books on the shelf
- one line containing n numbers $a_1, ..., a_n$ ($-150 \leq a_i \leq 150$) – the amounts by which the individual books increase or decrease your mood (the input order is the same as that on the shelf).

Output

Output the maximum mood you can reach as described above.

Sample input and output

Input	Output
3 71 -87 146	117
4 102 -150 -150 98	100
5 98 -150 -150 102 149	99

Problem C: Coasting

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

There is a big competition in one of Italy's many ski resorts. Participants ski down an entire mountainside that consists of a number of levels, connected by ski slopes, where the i-th and the $(i+1)$-th level are connected by precisely $(i + 1)$ slopes (which makes the ski resort overall triangular). Whenever a participant reaches the lower end of a slope (and therefore a new level) he has to decide whether to proceed with the left or right one of the two slopes starting near him.

Each slope is assigned a certain integer value (points) and a participant's score is calculated as the sum of the points on all the slopes he used in his descent. The competition starts at the mountaintop (level 0, where there is only one slope downhill) and ends when all participants have reached the lowest level (the one from which there are no slopes that lead further downhill). To win the competition you would have to reach the highest possible score, but to solve this problem it's sufficient to calculate it.

$$\begin{array}{c} \underline{1} \\ \underline{4}\ 3 \\ 5\ \underline{6}\ 7 \\ 8\ \underline{9}\ 0\ 9 \end{array}$$

Input

The input consists of

- one line containing N $(1 \leq N \leq 100)$ – the number of levels
- N lines that correspond to the slopes connecting the different levels, with the i-th line containing i numbers $a_{i,1}, a_{i,2}, ..., a_{i,i}$ $(|a_{i,k}| \leq 100,$ $1 \leq k \leq i)$ – the points on each slope.

Output

Output just one integer - the maximal amount of points that can be collected by a single participant.

Sample input and output

Input	Output
4 1 4 3 5 6 7 8 9 0 9	20

Problem D: Cannonballs

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Captain X always has some cannonballs on his ship to fend off pirates. Because he likes structure, he stores the cannonballs as pyramids. Each layer of those pyramids is an equilateral triangle filled with cannoballs. If a side of the bottom layer of a pyramid consists of n cannonballs, a side of the next layer consists of $n-1$ cannonballs, etc., up to the top layer which only contains 1 cannonball.

For example, a pyramid of size 3 consists of three layers:

```
  X

  X
 X X

  X
 X X
X X X
```

Obviously each of these triangles can only consist of 1, 3, 6, 10, etc. cannonballs. Therefore, a pyramid can only consist of 1, 4, 10, 20, etc. cannonballs.

Captain X is traveling and takes exactly m cannonballs with him. Find the minimal number of pyramids he has to build on his ship.

Input

The first line contains an integer T ($1 \leq T \leq 20$) – the number of testcases. Each testcase i consists of a single integer m_i ($1 \leq m_i \leq 3 \cdot 10^5$) – the number of cannon balls to be stacked into pyramids.

Output

For every testcase, output the minimal number of pyramids on a separate line.

Sample input and output

Input	Output
5 1 5 9 15 91	1 2 3 3 2

Problem E: Monster

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are a monster in a tower defense game. The game board is a rectangular grid. You can freely chose your starting point on the rightmost column. Every turn you can choose to move one field up, down or left. The player has placed towers on the board, that will decrease your HP every time you move into a field within their attack range (you will not take damage from spawning at your beginning position). The attack range of a tower is a square centered around it's position and a tower does not block you from entering the field it is placed on. You want to find the maximum amount of HP you can have left when reaching the leftmost column.

Input

The input consists of

- one line containing four integers N, M, T and H ($1 \leq N, M \leq 50$, $1 \leq T \leq 100$, $1 \leq H \leq 10^9$) – the number of columns and rows of the game board, the number of towers the player has placed and your starting HP
- T lines each containing four integers x_i, y_i, r_i and d_i ($1 \leq x_i \leq N$, $1 \leq y_i \leq M$, $0 \leq r_i \leq 5$, $1 \leq d_i \leq 1000$) – the column, row, range and damage of the i'th tower. A range of r means that the tower can attack a square area with side length $2r + 1$.

Output

Output just one integer - maximum HP that you can reach the leftmost column with. Output 0 if you can not reach the leftmost column.

Sample input and output

Input	Output
4 4 2 10 2 2 1 5 3 4 0 2	8
4 4 5 10 1 1 0 5 1 2 0 5 2 1 0 5 1 3 1 2 4 4 1 5	4
3 3 1 5 2 2 1 2	1
4 3 1 5 3 2 1 10	0

Problem F: Rabbit 2D

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Here is our lovely rabbit again, do you remember that fearless little guy? Thanks to your help he has now reached a bigger road: it's N squares long and 2 squares wide. And while there are no squares with grass and carrots anymore, there still are some with wolves on his way. Also the rabbit is very exhausted and can only jump on squares that have at least one common point (a corner or a side) with the one he is on. It's clear that he still can't jump on squares with wolves on them. Please help him to find out if it's possible for him to reach the rightmost top square from the leftmost top one.

Input

The input consists of

- one line containing an integer N ($2 \leq N \leq 50$) – the length of the road
- two lines each containing a string – the description of the road: '`W`' means there are wolves on that square, '.' is a clear square. The wolves promised to not stay at the first and last top squares.

Output

Output "YES" if the rabbit is able to reach the rightmost top square and "NO" otherwise.

Sample input and output

Input	Output
4 .W..	YES
4 .W.. ..W.	YES
7 .W..W.. ...WWW.	NO
2 .. WW	YES
7 .W.W.W. W.W.W.W	YES