Codeforces Round #579 (Div. 3)

A. Circle of Students

2 seconds, 256 megabytes

There are n students standing in a circle in some order. The index of the i-th student is p_i. It is guaranteed that all indices of students are distinct integers from 1 to n (i. e. they form a permutation).

Students want to start a round dance. A **clockwise** round dance can be started if the student 2 comes right after the student 1 in clockwise order (there are no students between them), the student 3 comes right after the student 2 in clockwise order, and so on, and the student n comes right after the student $n-1$ in clockwise order. A **counterclockwise** round dance is almost the same thing — the only difference is that the student i should be right after the student $i-1$ in counterclockwise order (this condition should be met for every i from 2 to n).

For example, if the indices of students listed in clockwise order are $[2, 3, 4, 5, 1]$, then they can start a clockwise round dance. If the students have indices $[3, 2, 1, 4]$ in clockwise order, then they can start a counterclockwise round dance.

Your task is to determine whether it is possible to start a round dance. Note that the students cannot change their positions before starting the dance; they cannot swap or leave the circle, and no other student can enter the circle.

You have to answer q independent queries.

Input

The first line of the input contains one integer q $(1 \le q \le 200)$ — the number of queries. Then q queries follow.

The first line of the query contains one integer n $(1 \le n \le 200)$ — the number of students.

The second line of the query contains a permutation of indices $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$, where p_i is the index of the i-th student (in clockwise order). It is guaranteed that all p_i are distinct integers from 1 to n (i. e. they form a permutation).

Output

For each query, print the answer on it. If a round dance can be started with the given order of students, print "YES". Otherwise print "NO".

input

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

output

```
YES
YES
NO
YES
YES
```

B. Equal Rectangles

2 seconds, 256 megabytes

You are given $4n$ sticks, the length of the i-th stick is a_i.

You have to create n rectangles, each rectangle will consist of exactly 4 sticks from the given set. The rectangle consists of four sides, opposite sides should have equal length and all angles in it should be right. Note that each stick can be used in only one rectangle. Each stick should be used as a side, you cannot break the stick or use it not to the full length.

You want to all rectangles to have equal area. The area of the rectangle with sides a and b is $a \cdot b$.

Your task is to say if it is possible to create exactly n rectangles of equal area or not.

You have to answer q independent queries.

Input

The first line of the input contains one integer q $(1 \le q \le 500)$ — the number of queries. Then q queries follow.

The first line of the query contains one integer n $(1 \le n \le 100)$ — the number of rectangles.

The second line of the query contains $4n$ integers $a_1, a_2, \ldots, a_{4n}$ ($1 \le a_i \le 10^4$), where a_i is the length of the i-th stick.

Output

For each query print the answer to it. If it is impossible to create exactly n rectangles of equal area using given sticks, print "NO". Otherwise print "YES".

input

```
5
1
1 1 10 10
2
10 5 2 10 1 1 2 5
2
10 5 1 10 5 1 1 1
2
1 1 1 1 1 1 1 1
1
10000 10000 10000 10000
```

output

```
YES
YES
NO
YES
YES
```

C. Common Divisors

2 seconds, 256 megabytes

You are given an array a consisting of n integers.

Your task is to say the number of such positive integers x such that x divides **each** number from the array. In other words, you have to find the number of common divisors of all elements in the array.

For example, if the array a will be $[2, 4, 6, 2, 10]$, then 1 and 2 divide each number from the array (so the answer for this test is 2).

Input

The first line of the input contains one integer n $(1 \le n \le 4 \cdot 10^5)$ — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$), where a_i is the i-th element of a.

Output

Print one integer — the number of such positive integers x such that x divides **each** number from the given array (in other words, the answer is the number of common divisors of all elements in the array).

input

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

output

```
1
```

input

```
6
6 90 12 18 30 18
```

output

```
4
```

D1. Remove the Substring (easy version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is the length of the string.

You are given a string s and a string t, both consisting only of lowercase Latin letters. It is guaranteed that t can be obtained from s by removing some (possibly, zero) number of characters (not necessary contiguous) from s without changing order of remaining characters (in other words, it is guaranteed that t is a subsequence of s).

For example, the strings "`test`", "`tst`", "`tt`", "`et`" and "" are subsequences of the string "`test`". But the strings "`tset`", "`se`", "`contest`" are not subsequences of the string "`test`".

You want to remove some substring (contiguous subsequence) from s of **maximum possible length** such that after removing this substring t will remain a subsequence of s.

If you want to remove the substring $s[l;r]$ then the string s will be transformed to $s_1 s_2 \dots s_{l-1} s_{r+1} s_{r+2} \dots s_{|s|-1} s_{|s|}$ (where $|s|$ is the length of s).

Your task is to find the maximum possible length of the substring you can remove so that t is still a subsequence of s.

Input

The first line of the input contains one string s consisting of at least 1 and at most 200 lowercase Latin letters.

The second line of the input contains one string t consisting of at least 1 and at most 200 lowercase Latin letters.

It is guaranteed that t is a subsequence of s.

Output

Print one integer — the maximum possible length of the substring you can remove so that t is still a subsequence of s.

input

```
bbaba
bb
```

output

```
3
```

input

```
baaba
ab
```

output

```
2
```

input

```
abcde
abcde
```

output

```
0
```

input

```
asdfasdf
fasd
```

output

```
3
```

D2. Remove the Substring (hard version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is the length of the string.

You are given a string s and a string t, both consisting only of lowercase Latin letters. It is guaranteed that t can be obtained from s by removing some (possibly, zero) number of characters (not necessary contiguous) from s without changing order of remaining characters (in other words, it is guaranteed that t is a subsequence of s).

For example, the strings "`test`", "`tst`", "`tt`", "`et`" and "" are subsequences of the string "`test`". But the strings "`tset`", "`se`", "`contest`" are not subsequences of the string "`test`".

You want to remove some substring (contiguous subsequence) from s of **maximum possible length** such that after removing this substring t will remain a subsequence of s.

If you want to remove the substring $s[l;r]$ then the string s will be transformed to $s_1 s_2 \dots s_{l-1} s_{r+1} s_{r+2} \dots s_{|s|-1} s_{|s|}$ (where $|s|$ is the length of s).

Your task is to find the maximum possible length of the substring you can remove so that t is still a subsequence of s.

Input

The first line of the input contains one string s consisting of at least 1 and at most $2 \cdot 10^5$ lowercase Latin letters.

The second line of the input contains one string t consisting of at least 1 and at most $2 \cdot 10^5$ lowercase Latin letters.

It is guaranteed that t is a subsequence of s.

Output

Print one integer — the maximum possible length of the substring you can remove so that t is still a subsequence of s.

input

```
bbaba
bb
```

output

```
3
```

input

```
baaba
ab
```

output

```
2
```

input

```
abcde
abcde
```

output

```
0
```

input

```
asdfasdf
fasd
```

output

```
3
```

E. Boxers

2 seconds, 256 megabytes

There are n boxers, the weight of the i-th boxer is a_i. Each of them can change the weight by no more than 1 before the competition (the weight cannot become equal to zero, that is, it must remain positive). Weight is always an integer number.

It is necessary to choose the largest boxing team in terms of the number of people, that all the boxers' weights in the team are different (i.e. unique).

Write a program that for given current values a_i will find the maximum possible number of boxers in a team.

It is possible that after some change the weight of some boxer is 150001 (but no more).

Input

The first line contains an integer n ($1 \le n \le 150000$) — the number of boxers. The next line contains n integers $a_1, a_2, \dots, a_n$, where a_i ($1 \le a_i \le 150000$) is the weight of the i-th boxer.

Output

Print a single integer — the maximum possible number of people in a team.

input

```
4
3 2 4 1
```

output

```
4
```

input

```
6
1 1 1 4 4 4
```

output

```
5
```

In the first example, boxers should not change their weights — you can just make a team out of all of them.

In the second example, one boxer with a weight of 1 can be increased by one (get the weight of 2), one boxer with a weight of 4 can be reduced by one, and the other can be increased by one (resulting the boxers with a weight of 3 and 5, respectively). Thus, you can get a team consisting of boxers with weights of $5, 4, 3, 2, 1$.

F1. Complete the Projects (easy version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is that you should complete all the projects in easy version but this is not necessary in hard version.

Polycarp is a very famous freelancer. His current rating is r units.

Some very rich customers asked him to complete some projects for their companies. To complete the i-th project, Polycarp needs to have at least a_i units of rating; after he completes this project, his rating will change by b_i (his rating will increase or decrease by b_i) (b_i can be positive or negative). Polycarp's rating should not fall below zero because then people won't trust such a low rated freelancer.

Is it possible to complete all the projects? Formally, write a program to check if such an order of the projects exists, that Polycarp has enough rating before starting each project, and he has non-negative rating after completing each project.

In other words, you have to check that there exists such an order of projects in which Polycarp will complete them, so he has enough rating before starting each project, and has non-negative rating after completing each project.

Input

The first line of the input contains two integers n and r ($1 \le n \le 100, 1 \le r \le 30000$) — the number of projects and the initial rating of Polycarp, respectively.

The next n lines contain projects, one per line. The i-th project is represented as a pair of integers a_i and b_i ($1 \le a_i \le 30000$, $-300 \le b_i \le 300$) — the rating required to complete the i-th project and the rating change after the project completion.

Output

Print "`YES`" or "`NO`".

input

```
3 4
4 6
10 -2
8 -1
```

output

```
YES
```

input

```
3 5
4 -5
4 -2
1 3
```

output

```
YES
```

input

```
4 4
5 2
5 -3
2 1
4 -2
```

output

```
YES
```

input

```
3 10
10 0
10 -10
30 0
```

output

```
NO
```

In the first example, the possible order is: $1, 2, 3$.

In the second example, the possible order is: $2, 3, 1$.

In the third example, the possible order is: $3, 1, 4, 2$.

F2. Complete the Projects (hard version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is that you should complete all the projects in easy version but this is not necessary in hard version.

Polycarp is a very famous freelancer. His current rating is r units.

Some very rich customers asked him to complete some projects for their companies. To complete the i-th project, Polycarp needs to have at least a_i units of rating; after he completes this project, his rating will change by b_i (his rating will increase or decrease by b_i) (b_i can be positive or negative). Polycarp's rating should not fall below zero because then people won't trust such a low rated freelancer.

Polycarp can choose the order in which he completes projects. Furthermore, he can even skip some projects altogether.

To gain more experience (and money, of course) Polycarp wants to choose the subset of projects **having maximum possible size** and the order in which he will complete them, so he has enough rating before starting each project, and has non-negative rating after completing each project.

Your task is to calculate the maximum possible size of such subset of projects.

Input

The first line of the input contains two integers n and r ($1 \le n \le 100, 1 \le r \le 30000$) — the number of projects and the initial rating of Polycarp, respectively.

The next n lines contain projects, one per line. The i-th project is represented as a pair of integers a_i and b_i ($1 \le a_i \le 30000$, $-300 \le b_i \le 300$) — the rating required to complete the i-th project and the rating change after the project completion.

Output

Print one integer — the size of **the maximum possible** subset (possibly, empty) of projects Polycarp can choose.

input

```
3 4
4 6
10 -2
8 -1
```

output

```
3
```

input

```
5 20
45 -6
34 -15
10 34
1 27
40 -45
```

output

```
5
```

input

```
3 2
300 -300
1 299
1 123
```

output

```
3
```