

Air Traffic Control II (atc2)

You may recall that Edoardo, who has recently been hired by the Municipality of Milan as an Air Traffic Control Operator, is working on the *paper-plane project*. He has already built T control towers and shot down every obstacle to ease the flight of the planes. However, the project is still not going well: there are too many other things in the air at the same time as the paper planes!

Edoardo has been working closely with the authorities in order to work out a solution, and the only option available is to pay for reserving some *air corridors* for the sole use of this project.

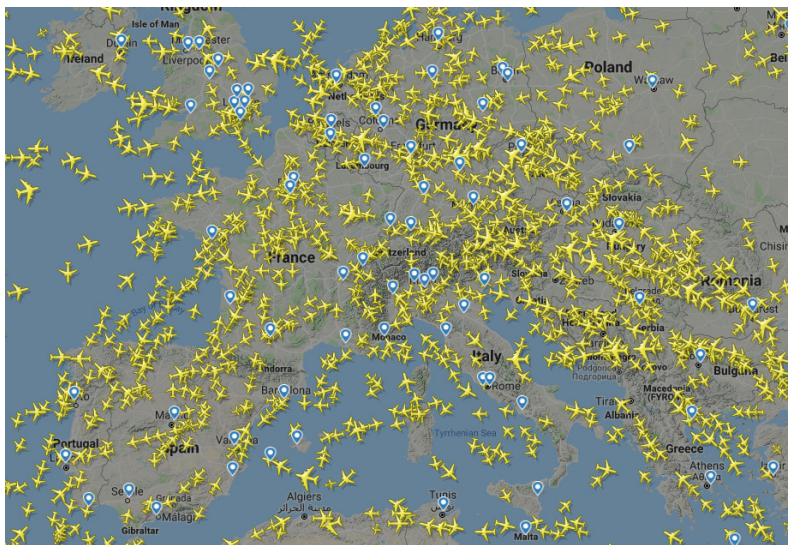


Figure 1: Real air traffic in Europe on a Friday morning (source: *FlightRadar24*).

Milan can be represented by a square grid of size $N \times N$, where each cell can be either:

- an empty space, represented with a “.” in input;
- a control tower, represented with a “T” in input.

The control tower placed on cell (i_1, j_1) can throw a paper plane towards any other control tower placed on cell (i_2, j_2) , using a dedicated air corridor. The reservation of such a corridor costs $(i_2 - i_1)^2 + (j_2 - j_1)^2$ euros.

Since booking air corridors is quite expensive, Edoardo wants to reserve only some of them, in such a way that makes possible for each tower to exchange paper planes with every other tower, possibly through some intermediate towers. What is the *minimum* cost that he has to pay?

Among the attachments of this task you may find a template file `atc2.*` with a sample incomplete implementation.

Input

The first line contains one integer N , the size of the grid. The next N lines contain N characters each. The first character of the first line represents position $(1, 1)$ of the grid. The last character of the last line represents position (N, N) .

Output






You need to write a single line with an integer: the minimum cost required to reserve the air corridors needed for the project.

Constraints

- $2 \leq N \leq 3000$.
- The number of control towers in the map is at least 2 and does not exceed 3000.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N = 2$.

- **Subtask 3** (45 points) $N \leq 1000$ and the number of control towers does not exceed 300.

- **Subtask 4** (30 points) $N \leq 1000$.

- **Subtask 5** (15 points) No additional limitations.


Examples

input	output
4 T T	18
5 . . . T T T	13

Explanation

In the **first sample case**, the only choice is to reserve an air corridor between $(1, 1)$ and $(4, 4)$ at cost $(4 - 1)^2 + (4 - 1)^2 = 18$.

In the **second sample case**, the cheapest choice is to reserve an air corridor between $(1, 4)$ and $(3, 3)$ at cost $(3 - 1)^2 + (3 - 4)^2 = 5$, and another air corridor between $(3, 3)$ and $(5, 5)$ at cost $(5 - 3)^2 + (5 - 3)^2 = 8$, for a total cost of 13.

Chief Executive Officer (ceo)

At *Luci di Nochia*, the most important company producing flash lights for smartphones, the proportions between bosses and employees are very regular.

Luca, the CEO, is the leader of the company. Under his control there are A_0 direct subordinates; under each of them there are other A_1 people, and so on. Generalizing, every person at level i has exactly A_i direct subordinates.

Luca, who is a bit greedy, is worried that this structure has led to a company with too many employees: he has even lost the count of them! Given all the values A_i for $i = 0 \dots N - 1$, count how many people work at his company.

👉 Among the attachments of this task you may find a template file `ceo.*` with a sample incomplete implementation.



Figure 1: Luca, while trying to count his employees with the calculator app of his smartphone.

Input

The first line contains the only integer N . The second line contains N integers A_i .

Output

You need to write a single line with an integer: the total number of employees Luca has, including himself.

Constraints

- $0 \leq N \leq 1000$.
- $1 \leq A_i \leq 1\,000\,000$ for each $i = 0 \dots N - 1$.
- The answer is at most 10^{18} .

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
🟡🟢🟣🟤
- **Subtask 2** (10 points) $N \leq 3$.
🟡🟢🟣🟤
- **Subtask 3** (10 points) The number of employees does not exceed 10^9 .
🟡🟢🟣🟤
- **Subtask 4** (15 points) $A_i = 2$ for all $i = 0 \dots N - 1$.
🟡🟢🟣🟤
- **Subtask 5** (15 points) $A_i = 3$ for all $i = 0 \dots N - 1$.
🟡🟢🟣🟤
- **Subtask 6** (20 points) $A_i = k$ for some value of k and for all $i = 0 \dots N - 1$.
🟡🟢🟣🟤
- **Subtask 7** (30 points) No additional limitations.
🟡🟢🟣🟤

Examples

input	output
2 2 3	9
4 4 1 2 1	25

Explanation

In the **first sample case** there are 9 people: Luca, two level-1 subordinates and six level-2 subordinates.

In the **second sample case** there are 25 people:

- Luca;
- 4 level-1 employees;
- 4 level-2 employees;
- 8 level-3 employees;
- 8 level-4 employees.

Long Columns (columns2)

There is no peace for the new OIS building: just after the grand opening, it has been seized due to alleged irregularities discovered in the building permits, and Edoardo is forced to start over once again with a minimal budget!

So far, he has retrieved N used prefabricated columns of length L_i meters. The construction company, however, is not able to cope with those columns and requires all of them to have a fixed length K .

Edoardo has already reached out a friend who is able to cut a column of length L_i into two parts: one of the desired length K and the other of length $L_i - K$.

In order to avoid difficulties as much as possible, Edoardo will always bring him *one* column, the *shortest* one (of length greater than K), to be cut. It is also known that after *ten* cuts an old column cannot withstand any more force and thus will not be cut further (but may still be used).

Respecting this courtesy rule (bringing every time the shortest column available), determine how many columns C of length K Edoardo can obtain and how many unused meters M of columns will be left after the process.



Figure 1: A *CNC* laser cutting machine.

👉 Among the attachments of this task you may find a template file `columns.*` with a sample incomplete implementation.

Input

The first line contains integers N and K . The second line contains N integers L_i .

Output





You need to write a single line with two integers separated by a space: respectively, the number C of columns of length K that can be obtained and the number of meters M of unused columns.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- $1 \leq K \leq 1000$.
- $1 \leq L_i \leq 1000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (15 points) $N = 1$.

- **Subtask 3** (45 points) $N \leq 1000$.

- **Subtask 4** (40 points) No additional limitations.


Examples

input	output
2 5 10 9	3 4
3 10 10 5 120	11 25

Explanation

In the **first sample case**, Edoardo first brings to cut the second column, obtaining one column of the desired length ($K = 5$) and a column of $9 - 5 = 4$ meters (which cannot be cut further as $4 < 5$). Then, Edoardo brings to cut the first column, obtaining two columns of the desired length. Overall, he has obtained three columns of length $K = 5$ and four meters are left unused.

In the **second sample case**, the first column already has the desired length and the second one is too short. Thus, Edoardo brings to cut the third column, obtaining two columns of length 10 and 110 meters. Then, brings again the column of 110 meters obtaining two columns of length 10 and 100 meters. The same process goes on until he is left (besides the columns of $K = 10$ meters) with a column of 20 meters which cannot be cut further as it has already been cut ten times. Overall, he has obtained eleven columns of length $K = 10$ and 25 meters are left unused.

Not Another Pet Fair (exhibitions2)

After the huge success of their first *pet fair*, Edoardo and Luca are already moving on to the organization of its second edition. Since they both felt an excessive competitive mood in the first edition, they decided to exchange responsibilities as a sign of peace.

Thus, this year Luca (huge fan of dogs) will decide the guides of *cat* exhibits, and conversely, Edoardo (strong supporter of cats) will decide the guides of *dog* exhibits. Of course, this means that all guides of cat exhibits will be dog supporters, and conversely guides of dog exhibits will be cat supporters.

As usual, the fair consists in N different exhibitions, each of them with a specific *awesomeness coefficient* A_i for $i = 0 \dots N - 1$, which can be:

- **positive**, to indicate a cat exhibit guided by a dog supporter; or
- **negative**, to indicate a dog exhibit guided by a cat supporter.

During the fair, M groups of tourists will arrive, starting from different exhibitions E_j for $j = 0 \dots M - 1$. The guides, after presenting their exhibition to the incoming group, will direct the group to another exhibition of their choice, or politely invite the group to leave the fair. The groups will continue to follow the instructions of the guides until they are kindly requested to leave.

Edoardo and Luca know very well that the incoming groups will be significantly more experienced for this second edition. In particular, they will not remember only the most awesome exhibit encountered after leaving, as for the first edition. Instead, they will leave with a precise feeling of how much a pet superseded the other, computed as the **signed sum** S_j of the awesomeness coefficients of every exhibit they encountered *at least once* during their visit (counted once regardless of how many times they were visited).

As usual, Edoardo and Luca have developed the best possible strategies for their sides and have coordinated tactics with their fellow guides accordingly: Edoardo's team will do their best to maximise S_j (that is, make S_j as biased towards cats as possible), while Luca's team will try to minimise S_j (that is, make S_j as biased towards dogs as possible). Calculate the feeling S_j that each incoming group will experience after their visit, knowing that every guide will follow the best possible strategy for their side!

📎 Among the attachments of this task you may find a template file `exhibitions2.*` with a sample incomplete implementation.

Input

The first line contains integers N and M . The second line contains N integers A_i . The third line contains M integers E_i .



Figure 1: Long-standing enemies confronting for the ultimate supremacy once more.

Output







You need to write a single line with M integers S_i , corresponding to the feeling experienced by each group, assuming a perfect strategy from both competing sides.

Constraints

- $1 \leq N, M \leq 500\,000$.
- $-10^9 \leq A_i \leq 10^9$, $A_i \neq 0$ for each $i = 0 \dots N - 1$.
- $0 \leq E_i \leq N - 1$ for each $i = 0 \dots M - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $N = 2$.

- **Subtask 3** (25 points) $N, M \leq 10$.

- **Subtask 4** (20 points) $N, M \leq 5000$.

- **Subtask 5** (15 points) $N \leq 5000$.

- **Subtask 6** (30 points) No additional limitations.


Examples

input	output
3 4 40 -20 10 2 0 1 0	10 30 20 30
4 2 10 -5 10 -15 3 2	-5 0

Explanation

In the **first sample case**, the first group arrives at exhibit 2 and is immediately sent back home by the dog-loving guide, scoring a result of $S_0 = 10$ which is the lowest achievable. The second group (as the last one) arrives at exhibit 0 instead, and is sent to exhibit 1 and then to exhibit 2, scoring a result of $S_1 = S_3 = 40 - 20 + 10 = 30$. The third group arrives at exhibit 1, and is sent to exhibit 0 before being invited to leave, scoring a result of $S_2 = -20 + 40 = 20$.

In the **second sample case**, the first group visits exhibits 3 and 0 or 2 before leaving, for a total $S_0 = -15 + 10 = -5$. The second group visits exhibits 2, 3, 0, 1 for a total $S_1 = 10 - 15 + 10 - 5 = 0$.

Gasoline Stations (gasoline)

After many days of uninterrupted studying, William needs to take a break: a romantic trip with his girlfriend from Milan to Pordenone, to visit the famous Saint Valentine Park!

During the travel, he will encounter N gas stations where he can buy some fuel for his car. William knows the price-per-liter P_i of each station and he has already estimated how much gasoline G_i at minimum he has to have at each station in order to reach the next one (and, at the last station, to arrive in Pordenone) safely. Being very wise, he brought with him many tanks so that he can buy as much gasoline as he wants.



Figure 1: To fill or not to fill, that is the question.

Help William plan how much he has to spend at *minimum*, knowing that he starts the journey from the first gas station without any fuel in the tanks.

👉 Among the attachments of this task you may find a template file `gasoline.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The second line contains N integers P_i , the price-per-liter at the i -th station. The third line contains N integers G_i , the amount of liters William has to have in the tanks to reach the next station (or Pordenone) from the i -th station.

Output

You need to write a single line with an integer: the minimum amount of money needed for the whole trip.

Constraints

- $1 \leq N \leq 1\,000\,000$.
- $1 \leq P_i, G_i \leq 1\,000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points) Examples.
🔥🔥🔥🔥🔥
- **Subtask 2** (10 points) $N \leq 10$.
🔥🔥🔥🔥🔥
- **Subtask 3** (10 points) All P_i are equal.
🔥🔥🔥🔥🔥
- **Subtask 4** (10 points) All G_i are equal.
🔥🔥🔥🔥🔥
- **Subtask 5** (30 points) $N \leq 1000$.
🔥🔥🔥🔥🔥
- **Subtask 6** (40 points) No additional limitations.
🔥🔥🔥🔥🔥

Examples

input	output
5 30 25 30 10 10 5 6 4 2 4	460
5 10 30 20 10 15 7 10 100 20 3	1400

Explanation

In the **first sample case**, an optimal solution is:

- Buy 5 liters at the first station spending $5 \cdot 30 = 150$.
- Buy 10 liters at the second station spending $10 \cdot 25 = 250$.
- Don't buy anything at the next station.
- Buy 6 liters at the next station $6 \cdot 10 = 60$.
- Don't buy anything at the last station.

The total amount is $150 + 250 + 60 = 460$.

In the **second sample case**, an optimal solution is to buy all the gasoline needed at the first station spending $140 \cdot 10 = 1400$.

Code Refactoring (refactor)

Giorgio is working on a boring research project, and he is just too lazy to actually write any code for it. Thus, he decided to simply copy from the internet other people's code and pass it off as his own.

As you may already know, the best way to use someone else's code without being detected is to **rename** some variables, functions or similar. That is exactly what Giorgio wants to do.

```

76     for (i = 0; i < ip->number_of_packets; i++) {
77         int offset;
78
79         for (offset = 0; USBTV_CHUNK_SIZE * offset < size; offset++)
80             usb_tv_image_chunk(usb_tv,
81                                 (__be32 *)&data[USBTV_CHUNK_SIZE * offset]);
82     }
    
```

The code that Giorgio wants to plagiarize is written in valid C, with some further assumptions (see the “Constraints” section for details). In order to perform the required replacement in this restricted version of C, it is enough to consider the syntax rules for identifiers, comments and string literals.

With the term **identifiers** we refer to variable names, function names and such. In general, an identifier is a sequence of characters that:

- has an underscore or a lowercase/uppercase letter as the first character;
- is only formed by digits, underscores, lowercase/uppercase letters;
- is not inside a **comment** or a **string literal**.

We define **comments** like this: when a line of text contains “//” then everything from that point until the end of that line is considered a comment. Moreover, when a line contains “/*” then everything from that point until the first occurrence of “*/” is considered a comment (note that this type of comment can span multiple lines).

Note also that earlier comments “neutralize” other comments inside them. So if you find the starting sequence of a comment **inside another comment** then you should ignore the newly starting one.

A **string literal** is defined as a sequence of characters enclosed by quotes (the " character). String literals cannot span multiple lines. If a comment starter sequence (// or /*) is found inside a string literal it should be considered as just text, not as the start of a comment.

You will be given a valid identifier S that Giorgio wants to change, as well as the entire source code (guaranteed to be valid) of the program that should be updated.

Help Giorgio by counting how many occurrences of the identifier S can be found in the code!

Among the attachments of this task you may find a template file `refactor.*` with a sample incomplete implementation.

Input

The first line contains a valid identifier S . The second line is empty. From the third line until the EOF character is found (i.e. until the end of the input) the source code of Giorgio's program can be found.

Output

You need to write a single line with one integer: the number of occurrences of the identifier S in the source code.

Constraints

- Macros (i.e. `#define` statements) are **never used** in the source code.
- The backslash and question mark characters, `\` and `?`, are **never used** in the source code.
- The identifiers used in the code can be up to 100 000 characters long.
- Numerical constants will always be expressed in decimal (i.e. using *digits only*).
- The size of Giorgio's source code will not exceed 5 MB.
- The identifier S that Giorgio wants to change will be between 4 and 100 000 characters long and **will never be** a keyword of the language (e.g. "while") or a word that could in any way be mistaken for a pre-defined word (e.g. "stdio").

Scoring

Your program will be tested against several test cases. Each test case solved will count towards the total score of your submission.

It is guaranteed that, for at least 80% of the test cases, the identifier S will not exceed 100 characters.

Examples

input	output
<pre>pippo #include <stdio.h> int main() { int x; int pippo = 3; pippo-=2; // decrement pippo char frase[] = "ciao pippo"; return pippo; }</pre>	3
<pre>somma #include <stdio.h> int somma(int x, int y) { return x + y; } int main() { int bisomma = 2*somma(13, 45); printf("somma %d", bisomma / 2); return 0; }</pre>	2

Explanation

In the **first sample case**, there are 3 occurrences of the variable `pippo` being used in the source code.

In the **second sample case**, we should not rename variable `bisomma` as it is a different identifier.

Nation Infrastructures (streets)

William is playing his favorite game on his smartphone, where he has to grow a nation from the grounds to the sky! In this game, you have to build cities, earn money and spend it to make your nation rich and powerful. Infrastructures are also very important, and developing streets should be kept in mind at all times... but William realized only now that he has built N cities and zero roads between them!



Figure 1: Some of the roads that William could have built.

This game is very realistic, thus, you cannot build a road between two random cities as the terrain may be too steep. William knows exactly which city could be connected to which.

As you may imagine, the more streets you build, the better. However, building a road is expensive: it costs 1 million dollars to each of the two city it connects, and each city has a limited budget of D_i million dollars available. Help William determine which streets to build in order to maximize the total number of roads, while keeping the costs within the budget of every city.

Among the attachments of this task you may find a template file `streets.*` with a sample incomplete implementation.

Input

The first line contains the integers N and M , the number of cities and the number of possible roads. The second line contains N integers D_i , the budget of the i -th city. The next M lines contain a pair of integers each, representing the fact that a road may be built between those cities.

Output

You need to write K lines with a pair of integers each: the list of streets to build, in any order.

Constraints

- $2 \leq N \leq 100$.
- $1 \leq M \leq 5000$.
- $1 \leq D_i \leq 100$ for each $i = 0 \dots N - 1$.
- All the streets are bidirectional, there is at most one possible street for every pair of cities.
- No city can be connected to itself.
- It is possible to spend all the money.

Scoring

Your program will be tested against several test cases. The total score will be proportional to the sum of the scores obtained in each test case. The score for a test case is computed as follows:

$$\frac{1}{X + 1}$$

where X is the total unused budget (in million dollars) after building the roads.

Examples

input	output
5 8 3 2 1 3 1 0 1 0 3 0 4 1 2 1 3 2 3 2 4 3 4	1 0 0 4 3 0 2 3 1 3
6 10 1 1 5 1 1 1 2 1 2 0 2 3 2 4 2 5 1 0 0 3 3 4 4 5 5 0	1 0 3 4 2 5

Explanation

In the **first sample case**, if you build the reported streets you will spend all the budget, maximising the number of roads built.

In the **second sample case**, the solution is not perfect, having built 3 roads for total 6 million dollars. A better solution is to connect city 2 with all the others, exhausting the whole budget.

Full-Body Workout (workout)

Giorgio has recently undertaken a tough full-body workout plan, consisting of a sequence of N exercises, each with a *fatiguing coefficient* F_i for $i = 0 \dots N - 1$. In particular, some exercises are tiresome ($F_i > 0$) while others are restful ($F_i < 0$).

The plan has already proven to be effective, however, it takes up a lot of time every day! In order to save at least a bit of time, Giorgio has decided to completely remove the cool down times between the different exercises... and now needs to carefully choose the exercise scheduling, in order to keep the fatigue limited and be able to complete the workout plan anyway!



Figure 1: Giorgio after few months of the workout plan.

Suppose that the exercises are arranged in some order $F_{\pi_0}, F_{\pi_1}, \dots, F_{\pi_{N-1}}$ where $\pi_0 \dots \pi_{N-1}$ is a *permutation* of the numbers $0 \dots N - 1$. Then, the total fatigue values T_i reached from the training start until the end are obtained by summing the fatiguing coefficient, while avoiding to go below zero. In formulas:

$$\begin{array}{ll}
 \text{start:} & T_0 = 0 \\
 \text{after the first:} & T_1 = \max(F_{\pi_0}, 0) \\
 \text{after the second:} & T_2 = \max(T_1 + F_{\pi_1}, 0) \\
 \text{after the third:} & T_3 = \max(T_2 + F_{\pi_2}, 0) \\
 & \dots \\
 \text{end:} & T_N = \max(T_{N-1} + F_{\pi_{N-1}}, 0)
 \end{array}$$

Help Giorgio find a suitable arrangement π of exercises, such that the *maximum* of these T_i values is as small as possible!

Among the attachments of this task you may find a template file `workout.*` with a sample incomplete implementation.

Input

The first line contains the only integer N . The second line contains N integers F_i .

Output

You need to write two lines. The first line contains a single integer, the maximum fatigue reached during the training. The second line contains N integers, representing the permutation F_{π_i} of the numbers F_i in input achieving the declared fatigue value.

Constraints







- $1 \leq N \leq 20\,000$.
- $-100\,000 \leq F_i \leq 100\,000$ for each $i = 0 \dots N - 1$.

Scoring

Your program will be tested against several test cases grouped in subtasks. Your score on a subtask will be equal to the minimum score for one of its testcases multiplied by the value of the subtask. The score of a test case is computed as follows:

$$\min\left(\frac{T_{\text{cor}}^2}{T_{\text{out}}^2}, 1\right)$$

where T_{out} is the maximum fatigue of your solution, and T_{cor} is the maximum fatigue of our solution.

- **Subtask 1** (0 points) Examples.

- **Subtask 2** (10 points) $F_i \geq 0$ for each $i = 0 \dots N - 1$.

- **Subtask 3** (15 points) $F_i \geq -1$ for each $i = 0 \dots N - 1$.

- **Subtask 4** (30 points) $N \leq 10$.

- **Subtask 5** (25 points) $N \leq 1000$.

- **Subtask 6** (20 points) No additional limitations.


Examples

input	output
3 -30 10 10	10 10 -30 10
5 11 18 -30 21 -10	21 21 -30 11 -10 18

Explanation

In the **first sample case**, there is no better solution since after exercise 1 of fatigue $F_1 = 10$ the total fatigue is necessarily at least 10. In fact, maintaining the exercises in the original order would achieve a worse maximum fatigue of 20, as the values T_i would be 0, 0, 10, 20.

In the **second sample case**, there is no better solution since after exercise 3 of fatigue $F_3 = 21$ the total fatigue is necessarily at least 21. In fact, maintaining the exercises in the original order would achieve a worse maximum fatigue of 29, as the values T_i would be 0, 11, 29, 0, 21, 11.