# A. Double Cola

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sheldon, Leonard, Penny, Rajesh and Howard are in the queue for a "Double Cola" drink vending machine; there are no other people in the queue. The first one in the queue (Sheldon) buys a can, drinks it and doubles! The resulting two Sheldons go to the end of the queue. Then the next in the queue (Leonard) buys a can, drinks it and gets to the end of the queue as two Leonards, and so on. This process continues ad infinitum.

For example, Penny drinks the third can of cola and the queue will look like this: Rajesh, Howard, Sheldon, Sheldon, Leonard, Leonard, Penny, Penny.

Write a program that will print the name of a man who will drink the $n$-th can.

Note that in the very beginning the queue looks like that: Sheldon, Leonard, Penny, Rajesh, Howard. The first person is Sheldon.

## Input

The input data consist of a single integer $n$ ($1 \le n \le 10^9$).

It is guaranteed that the pretests check the spelling of all the five names, that is, that they contain all the five possible answers.

## Output

Print the single line — the name of the person who drinks the $n$-th can of cola. The cans are numbered starting from 1. Please note that you should spell the names like this: "`Sheldon`", "`Leonard`", "`Penny`", "`Rajesh`", "`Howard`" (without the quotes). In that order precisely the friends are in the queue initially.

### Examples

input
```
1
```
output
```
Sheldon
```

input
```
6
```
output
```
Sheldon
```

input
```
1802
```
output
```
Penny
```

# B. Sets

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Vasya likes very much to play with sets consisting of positive integers. To make the game more interesting, Vasya chose $n$ non-empty sets in such a way, that no two of them have common elements.

One day he wanted to show his friends just how interesting playing with numbers is. For that he wrote out all possible unions of two different sets on $n \cdot (n - 1) / 2$ pieces of paper. Then he shuffled the pieces of paper. He had written out the numbers in the unions in an arbitrary order.

For example, if $n = 4$, and the actual sets have the following form $\{1, 3\}$, $\{5\}$, $\{2, 4\}$, $\{7\}$, then the number of set pairs equals to six. The six pieces of paper can contain the following numbers:

- $2, 7, 4.$
- $1, 7, 3;$
- $5, 4, 2;$
- $1, 3, 5;$
- $3, 1, 2, 4;$
- $5, 7.$

Then Vasya showed the pieces of paper to his friends, but kept the $n$ sets secret from them. His friends managed to calculate which sets Vasya had thought of in the first place. And how about you, can you restore the sets by the given pieces of paper?

## Input

The first input file line contains a number $n$ $(2 \le n \le 200)$, $n$ is the number of sets at Vasya's disposal. Then follow sets of numbers from the pieces of paper written on $n \cdot (n - 1) / 2$ lines. Each set starts with the number $k_i$ $(2 \le k_i \le 200)$, which is the number of numbers written of the $i$-th piece of paper, and then follow $k_i$ numbers $a_{ij}$ $(1 \le a_{ij} \le 200)$. All the numbers on the lines are separated by exactly one space. It is guaranteed that the input data is constructed according to the above given rules from $n$ non-intersecting sets.

## Output

Print on $n$ lines Vasya's sets' description. The first number on the line shows how many numbers the current set has. Then the set should be recorded by listing its elements. Separate the numbers by spaces. Each number and each set should be printed exactly once. Print the sets and the numbers in the sets in any order. If there are several answers to that problem, print any of them.

It is guaranteed that there is a solution.

**Examples**

input

```
4
3 2 7 4
3 1 7 3
3 5 4 2
3 1 3 5
4 3 1 2 4
2 5 7
```

output

```
1 7
2 2 4
2 1 3
1 5
```

input

```
4
5 6 7 8 9 100
4 7 8 9 1
4 7 8 9 2
3 1 6 100
3 2 6 100
2 1 2
```

output

```
3 7 8 9
2 6 100
1 1
1 2
```

input

```
3
2 1 2
2 1 3
2 2 3
```

output

```
1 1
1 2
1 3
```

# C. General Mobilization

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Berland Kingdom is a set of $n$ cities connected with each other with $n - 1$ railways. Each road connects exactly two different cities. The capital is located in city $1$. For each city there is a way to get from there to the capital by rail.

In the $i$-th city there is a soldier division number $i$, each division is characterized by a number of $a_i$. It represents the priority, the smaller the number, the higher the priority of this division. All values of $a_i$ are different.

One day the Berland King Berl Great declared a general mobilization, and for that, each division should arrive in the capital. Every day from every city except the capital a train departs. So there are exactly $n - 1$ departing trains each day. Each train moves toward the capital and finishes movement on the opposite endpoint of the railway on the next day. It has some finite capacity of $c_j$, expressed in the maximum number of divisions, which this train can transport in one go. Each train moves in the direction of reducing the distance to the capital. So each train passes exactly one railway moving from a city to the neighboring (where it stops) toward the capital.

In the first place among the divisions that are in the city, division with the smallest number of $a_i$ get on the train, then with the next smallest and so on, until either the train is full or all the divisions are be loaded. So it is possible for a division to stay in a city for a several days.

The duration of train's progress from one city to another is always equal to $1$ day. All divisions start moving at the same time and end up in the capital, from where they don't go anywhere else any more. Each division moves along a simple path from its city to the capital, regardless of how much time this journey will take.

Your goal is to find for each division, in how many days it will arrive to the capital of Berland. The countdown begins from day $0$.

## Input

The first line contains the single integer $n$ ($1 \le n \le 5000$). It is the number of cities in Berland. The second line contains $n$ space-separated integers $a_1, a_2, \ldots, a_n$, where $a_i$ represents the priority of the division, located in the city number $i$. All numbers $a_1, a_2, \ldots, a_n$ are different ($1 \le a_i \le 10^9$). Then $n - 1$ lines contain the descriptions of the railway roads. Each description consists of three integers $v_j, u_j, c_j$, where $v_j, u_j$ are number of cities connected by the $j$-th rail, and $c_j$ stands for the maximum capacity of a train riding on this road ($1 \le v_j, u_j \le n, v_j \ne u_j, 1 \le c_j \le n$).

## Output

Print sequence $t_1, t_2, \ldots, t_n$, where $t_i$ stands for the number of days it takes for the division of city $i$ to arrive to the capital. Separate numbers with spaces.

### Examples

**input**

```
4
40 10 30 20
1 2 1
2 3 1
4 2 1
```

**output**

```
0 1 3 2
```

**input**

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

**output**

```
0 1 4 2 3
```

# D. Two out of Three

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya has recently developed a new algorithm to optimize the reception of customer flow and he considered the following problem.

Let the queue to the cashier contain $n$ people, at that each of them is characterized by a positive integer $a_i$ — that is the time needed to work with this customer. What is special about this very cashier is that it can serve two customers simultaneously. However, if two customers need $a_i$ and $a_j$ of time to be served, the time needed to work with both of them customers is equal to $max(a_i, a_j)$. Please note that working with customers is an uninterruptable process, and therefore, if two people simultaneously come to the cashier, it means that they begin to be served simultaneously, and will both finish simultaneously (it is possible that one of them will have to wait).

Vasya used in his algorithm an ingenious heuristic — as long as the queue has more than one person waiting, then *some two people of the first three standing in front of the queue are sent simultaneously*. If the queue has only one customer number $i$, then he goes to the cashier, and is served within $a_i$ of time. Note that the total number of phases of serving a customer will always be equal to $\lceil n / 2 \rceil$.

Vasya thinks that this method will help to cope with the queues we all hate. That's why he asked you to work out a program that will determine the minimum time during which the whole queue will be served using this algorithm.

## Input

The first line of the input file contains a single number $n$ ($1 \le n \le 1000$), which is the number of people in the sequence. The second line contains space-separated integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^6$). The people are numbered starting from the cashier to the end of the queue.

## Output

Print on the first line a single number — the minimum time needed to process all $n$ people. Then on $\lceil n / 2 \rceil$ lines print the order in which customers will be served. Each line (probably, except for the last one) must contain two numbers separated by a space — the numbers of customers who will be served at the current stage of processing. If $n$ is odd, then the last line must contain a single number — the number of the last served customer in the queue. The customers are numbered starting from $1$.

### Examples

**input**

```
4
1 2 3 4
```

**output**

```
6
1 2
3 4
```

**input**

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

**output**

```
8
1 3
2 5
4
```

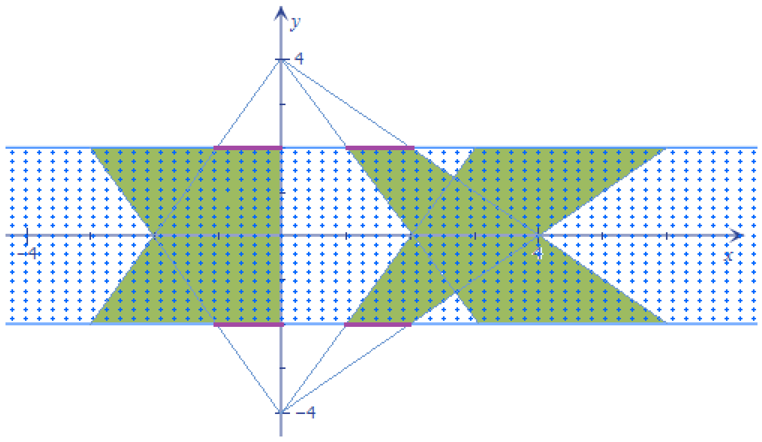
# E. Corridor

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a house plan.

Let the house be represented by an infinite horizontal strip defined by the inequality $-h \le y \le h$. Strictly outside the house there are two light sources at the points $(0, f)$ and $(0, -f)$. Windows are located in the walls, the windows are represented by segments on the lines $y = h$ and $y = -h$. Also, the windows are arranged symmetrically about the line $y = 0$.

Your task is to find the area of the floor at the home, which will be lighted by the sources of light.



## Input

The first line of the input file contains three integers $n$, $h$ and $f$ $(1 \le n \le 500,\ 1 \le h \le 10,\ h < f \le 1000)$. Next, $n$ lines contain two integers each $l_i$, $r_i$ $(-5000 \le l_i < r_i \le 5000)$, each entry indicates two segments. Endpoints of the first segment are $(l_i, h)$-$(r_i, h)$, and endpoints of the second segment are $(l_i, -h)$-$(r_i, -h)$. These segments describe location of windows. Numbers in the lines are space-separated. It is guaranteed that no two distinct segments have common points.

## Output

Print the single real number — the area of the illuminated part of the floor with an absolute or relative error of no more than $10^{-4}$.

### Examples

input

```
1 1 2
-1 1
```

output

```
10.0000000000
```

input

```
2 2 4
-1 0
1 2
```

output

```
23.3333333333
```

## Note

The second sample test is shown on the figure. Green area is the desired area of the illuminated part of the floor. Violet segments indicate windows.