**Welcome to**

# Santa's Algorithmic Challenge

---

Good Luck

Have a great time!

*Santa and the organizers*

# A. Claussian

| | |
|---|---|
| Time limit | 1 second |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

Finally, you have arrived at this year's Santa's Algorithmic Challenge and opened the first problem – reportedly the easiest. Hard to believe, it's already asking you to compute some bizarre mathematical quantity (which is supposedly going to help Santa)!
For a string $s$, we define the *Claussian* of $s$ as the number of substrings of $s$ equal to `Santa`. Claussians have applications in automatic processing of letters from children.

You are given a string $s$ for which Santa would like you to compute the Claussian. If you do, you might get a present...

## Input format

The first (and only) line of the input contains a single string $s$ of length at most $100\ 000$. Each character of$s$ is either a lowercase or an uppercase letter of the English alphabet.

## Output format

Print a single integer – the Claussian of $s$.

### Sample 1

| Input | Output |
|---|---|
| `santa` | `0` |

## Sample 2

| Input | Output |
|---|---|
| `Santa` | `1` |

## Sample 3

| Input | Output |
|---|---|
| `SANTASantASaantasantasANTA` | `0` |

## Sample 4

| Input | Output |
|---|---|
| `kyrHASantaSantafuSantasantaogjSantaap` | `4` |

## Notes

Substrings are contiguous subsequences of a string, and lower/upper case matters; for example, the string `abc` contains `bc` as a substring, but not `ac` nor `Bc` nor `ca`.

# Presents

| | |
|---|---|
| Time limit | 0.5 seconds |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

Santa Claus is coming to town again. Today he is paying a visit to the greater Lausanne area, which comprises Lausanne city (zone 11) and the suburbs (zone 12 – including EPFL). (What's up with the zones, you ask? Well, as you cross the Swiss border, even reindeer services suddenly become expensive, and Santa sometimes resorts to using public transport.)
Needless to say, Santa will be bringing presents to the residents of both zones. He strives for fairness above all – he wants to leave exactly the same number of presents in each zone. And this number better be positive!
Santa's presents have been prepackaged by his elves into $n$ bags. Since last year's Algorithmic Challenge, the elves have taken up interest in computer science. In consequence, now the number of presents in every bag is a power of two.

Santa wants to leave some of the bags in zone 11 and some in zone 12. He can also take some bags back with him, if there are any left. Because the bags have been so meticulously packed by the elves, he does not want to open any bag and distribute its contents between the zones.

He is now asking you: is it possible to distribute some of the bags into the two zones so that each zone receives the same positive number of presents?

## Input format

The first line of the input contains a single integer $n$ – the number of bags Santa has at his disposal ($1 \leq n \leq 30$). The second line contains $n$ space-separated integers $a_1, a_2, ..., a_n$, where $a_i$ is the number of presents in the $i$-th bag. For all $i = 1, ..., n$ it holds that $1 \leq a_i \leq 2^{30}$ and that $a_i$ is a power of two.

## Output format

Print a single word, the answer to Santa's question: `YES` or `NO`.

## Sample

### Input

```
4

1 2 1073741824 1
```

### Output

```
YES
```

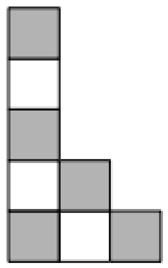
## Notes

In the sample test case, Santa can leave the two bags containing one present each in zone 11, and the bag containing two presents in zone 12. Unfortunately, he will have to return with the huge bag.

# Chessboard

| | |
|---|---|
| Time limit | 1 second |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

This year you received a very strange present from Santa – a modified chessboard. It is made from separate vertical chessboard lines, aligned by their bottoms and sorted in non-ascending-height order.



You have failed in your attempts to devise a reasonable two-player game to be played on this board, and now you're just wondering: what is the minimum number of rooks needed to attack all cells of such a chessboard? Answer this question, and give an example of such a minimum-size set of rooks.

## Input format

The first line of the input holds a single integer $n$ – the number of vertical chessboard lines ($1 \le n \le 1000$). The second line of the input holds $n$ space-separated integers $a_1, a_2, \dots, a_n$ – number of cells in each line ($1 \le a_i \le 1000$, $a_1 \ge a_2 \ge \dots \ge a_n$).

## Output format

In the first line print an integer: the minimum number of rooks $r$. Then print a set of rooks of size $r$ such that every cell of the chessboard is attacked by at least one rook. Specifically, in each of the next $r$ lines, print two space-separated integers: the coordinates of a rook (first the line number, then the number of cell in this line). Lines are 1-numbered from left to right, and cells are 1-numbered from bottom to top. If there are multiple such possible sets, print any of them.

## Sample 1

### Input

```
3
5 2 1
```

### Output

```
2
1 1
2 2
```

## Sample 2

### Input

```
1
1
```

### Output

```
1
1 1
```

## Sample 3

### Input

```
2
1 1
```

### Output

```
1
1 1
```

## Notes

A rook attacks each cell of the board which is in the same row or in the same column as the rook. In particular, it attacks the cell on which it is standing.

# Santa's Bag

|  |  |
|---|---|
| Time limit | 1 second |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

Santa is preparing a bag of presents for all the students of EPFL that he has claussified as naughty. (The group is surprisingly large this year.) The naughty students are not supposed to receive valuable presents. On the contrary, Santa wants to teach them a lesson and has already selected several series of undesirable items. His collection of punitive presents includes old socks, magazines with only advertisements, as well as some very large but empty boxes.
There are $n$ series of presents prepared. One item from the $i$-th series has size $s_i$ and value $v_i$. There is an unlimited supply of items from each of the series. Santa's bag has a total capacity of $m$, and he insists on filling all of it (not doing so would be a waste of resources, and in Switzerland that just doesn't go). Thus, a collection of items can be put in the bag if the sum of sizes of the items is exactly $m$.
Help Santa and compute the **minimum** total value of the bag.

## Input format

The first line of the input contains two space-separated integers $n$ and $m$ – the number of series and the capacity of Santa's bag ($1 \leq n, m \leq 200$). Each of the next $n$ lines contains two space-separated integers$s_i$ and $v_i$, which describe one series of presents ($1 \leq s_i, v_i \leq 200$).

## Output format

Output the minimum value of the bag. You may assume that it is always possible to fill the bag.

## Sample 1

### Input

```
5 100
1 1
2 2
3 3
4 4
5 4
```

### Output

```
80
```

## Sample 2

### Input

```
4 5
1 10
2 3
3 4
4 1
```

### Output

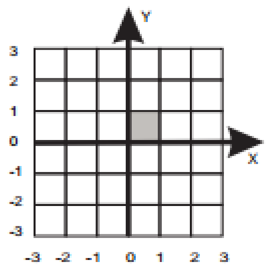
```
7
```

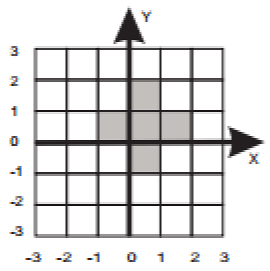
# Cameras

| | |
|---|---|
| Time limit | 1.5 seconds |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

Everybody knows Santa Claus – the mysterious benevolent figure who runs a large gift-giving operation out of his base in Lapland. It is much less known that the most important building in the headquarters is the R&D center, where Santa keeps his top-secret algorithms for solving fairest-child-present-allocation problems.
What the legend doesn't say at all is that Sonta Clausson is lurking in the shadows – a cunning impersonator who is plotting to enter Lapland from the west and then infiltrate the building, dressed up as Santa, learn Santa's secret algorithms, and analyze their performance.

Fortunately, Santa knows that the algorithms are in danger, and he is taking measures to defend the headquarters. He is installing fingerprint readers at all entrances and cameras alongside all walls of the building. And he is asking you for help in estimating the cost of the latter.
Santa's R&D center, seen from above, is a complex made up of $n$ unit-length squares. We adopt a coordinate system where every square has sides parallel to the x-axis and the y-axis. The building forms a connected set of squares.





The number of cameras that Santa will install is equal to the perimeter of the building (the length of all walls). Compute this number and help protect the algorithms from Sonta!

## Input format

First line of the input holds a single integer $n$ ($1 \leq n \leq 100\,000$) – the number of single squares making up the building.
Then $n$ lines follow, each containing two space-separated integers $x_i$ and $y_i$ – the coordinates of the lower-left corner of the $i$-th square (satisfying $|x_i|, |y_i| \leq 10^9$). All given squares will be different.

## Output format

Output a single integer – the perimeter of the building.

### Sample 1

**Input**

```
1
0 0
```

**Output**

```
4
```

### Sample 2

**Input**

```
5
0 0
0 -1
-1 0
1 0
0 1
```

**Output**

```
12
```

## Sample 3

**Input**

```
6
0 0
1 0
2 0
2 1
2 2
1 2
```

**Output**

```
14
```

## Sample 4

**Input**

```
8
0 0
0 1
0 2
1 0
1 2
2 0
2 1
2 2
```

**Output**

```
16
```

## Notes

The building may have "holes" (see the last sample case). You should compute the entire perimeter, including that of the holes.

# Sequence

| | |
|---|---|
| Time limit | 3 seconds |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

By solving the task *Cameras* you have helped Santa secure his headquarters (containing Santa's priceless, top-secret algorithms). Unfortunately, fingerprint readers and surveillance cameras were no match for the villain Sonta Clausson – he ended up outsmarting everyone and getting into the building through the chimney.

Now Sonta is inside, and just one step away from learning the precious algorithms. They are stored in a safe which has a strange combination lock, and Sonta needs to break this lock to open the safe.

The current state of the lock is represented by a sequence of $n$ integer numbers $a_1, a_2, ..., a_n$. Sonta can change this sequence one number at a time – specifically, it takes Sonta one second to increment or decrement by one the current value of any number in the lock sequence. In other words, transforming the initial sequence into an integer sequence $b_1, b_2, ..., b_n$ takes Sonta $\sum_{i=1}^{n} |a_i - b_i|$ seconds.

The safe will open as soon as the sequence becomes *$K$-Laussian*. For an integer $K \geq 0$, we say that a sequence $b_1, b_2, ..., b_n$ is $K$-Laussian if the number of entries in the sequence which differ from the previous entry is at most $K$. More precisely, the number of indexes $i$, with $2 \leq i \leq n$, such that $b_{i-1} \neq b_i$, should be at most $K$.

The cunning Sonta will of course use the optimal way (the minimum possible time) to make the sequence$K$-Laussian and break the lock. Compute this time (in seconds) and find out if Santa will return in time to catch Sonta red-handed.

## Input format

In the first line of the input two space separated-integers $n$ and $K$ are given ($1 \leq n \leq 1000, 0 \leq K \leq 100$).

The second line contains $n$ space-separated integers $a_1, a_2, ..., a_n$ – the initial state of the lock ($0 \leq a_i \leq 10^9$).

## Output format

Output one integer – the minimum time in seconds (as described above).

### Sample 1

**Input**

```
5 2

1 2 3 4 5
```

**Output**

```
2
```

### Sample 2

**Input**

```
5 0

1 2 3 4 5
```

**Output**

```
6
```

### Sample 3

**Input**

```
5 1

0 0 0 0 0
```

**Output**

```
0
```

## Notes

- For example, in the first sample case, Sonta can change the sequence to `1 3 3 3 5` or to `2 2 3 4 4`; both are $2$-Laussian.

- Warning: the output value might not fit in a 32-bit integer.
- Warning: we only guarantee the solvability of this task in C++ and Java (under the given time limits).

# Reind Air

| | |
|---|---|
| Time limit | 2 seconds |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

The Laplandish airline Reind Air has hired you to analyze its route network. The network is a collection of direct two-way flights between airports. Arbitrarily long routes can be built from these direct flights by connecting at zero, one or more airports. Travel (possibly with connections) is currently possible between any two airports. Airports are numbered from $1$ to $n$.
Reind Air is worried about the connectivity of its network, and about existence of single points of failure. They are asking you questions of the following format: supposing that there is a failure at an airport $i$ and that all flights from and to $i$ are removed from the network, how many pairs of airports $(j,k)$ would become disconnected from each other? In other words, for an airport $i$, how many pairs $(j,k)$ are there such that every route from $j$ to $k$ goes through $i$?

## Input format

The first line of the input contains two space-separated integers $n$ and $m$ – the number of airports and the number of direct two-way flights, respectively ($1 \le n \le 20\ 000$, $1 \le m \le 200\ 000$).
The following $m$ lines describe the flights. Each line contains two space-separated integers $a_i$ and $b_i$ – airports served by the flight ($1 \le a_i, b_i \le n$, $a_i \ne b_i$).

## Output format

Output $n$ lines. The $i$-th line of the output should contain the number of pairs of airports $(j,k)$ for the airport$i$ as described above. You should count unordered pairs (for example, count only one of $(1,2)$, $(2,1)$).

## Sample 1

**Input**

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

**Output**

```
18
6
6
6
6
6
6
```

## Sample 2

**Input**

```
2 1
1 2
```

**Output**

```
1
1
```

## Sample 3

**Input**

```
3 2
1 2
```

**Output**

```
2
3
```

| Input | Output |
|---|---|
| 2 3 | 2 |

## Notes

Note that $j$ (or $k$) may be equal to $i$.

# Catching Santa

| | |
|---|---|
| Time limit | 1 second |
| Memory limit | 256Mb |
| Input | standard input |
| Output | standard output |

You already know that Santa will be bringing presents to EPFL. He is known to be very stealhy, but maybe, with all the engineering resources of EPFL at our disposal, we could be able to catch him and finally have a conclusive proof that he is real? Challenge accepted!
Santa will come to Lausanne by train, take some route through the city using public transport, and end up at EPFL. Our brilliant plan is to produce several Santa-detectors (the needed technology is almost ready) and place them at some bus and metro stops in the city. We will succeed if Santa ever comes near any of these stops.

Of course, the challenge is that we don't know what route through the city Santa is going to take, and we can't afford to place detectors at every minor bus stop. On the other hand, we are also not going to place any detector at the train station or at EPFL, because that would be too easy.

What is the minimum number of stops at which we must place detectors so that Santa must visit at least one of these stops on any route between the station and EPFL?

## Input format

First line of the input holds a single integer $n$ ($1 \leq n \leq 100$) – the number of bus/metro stops. All stops are numbered from $1$ to $n$.
Then $n$ lines follow. The $i$-th line describes connections of the $i$-th stop – the places that Santa can get to from the $i$-th stop. The line starts with an integer $K_i$ – the number of connections. Then $K_i$ space-separated connections follow. Each connection is either another stop (a number between $1$ and $n$) or the train station (`station`) or EPFL (`school`).
All connections are two-way. That is, if there is a connection from stop $i$ to stop $j$, then there is a connection from stop $j$ to stop $i$; moreover, if there is a connection from stop $i$ to `station`, then Santa is also able to get from the train station to stop $i$.

It is guaranteed that the train station and EPFL are connected (but only indirectly – through at least one stop).

## Output format

Output a single integer – the size of the smallest set of bus/metro stops with the property that, on any route from `station` to `school`, there is at least one stop from this set.

### Sample 1

**Input**

```
2
2 school station
2 station school
```

**Output**

```
2
```

### Sample 2

**Input**

```
3
3 school 2 3
3 school 1 3
3 station 1 2
```

**Output**

```
1
```

### Sample 3

**Input**

```
15
4 4 7 10 15
3 4 6 9
```

**Output**

```
4
```

**Input** **Output**

```
5 5 6 12 13 station

7 1 2 5 10 12 13 15

7 3 4 6 9 11 14 15

10 school 2 3 5 7 8 9 10 13 station

4 school 1 6 13

6 6 10 11 12 13 14

5 school 2 5 6 10

6 1 4 6 8 9 13

3 5 8 13

6 3 4 8 13 15 station

8 3 4 6 7 8 10 11 12

5 school 5 8 15 station

6 1 4 5 12 14 station
```