# Problem A. Movies

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

Deadpool is watching movies about Wolverine from different universes on a multiverse movie website. It is known that there are a total of $n$ movies, and each page of the website displays $k$ different movies (with the last page showing all remaining ones). Deadpool doesn't have much time, so he goes straight to the last page and watches all the movies from it.

Each movie lasts exactly $c$ minutes. How much time did he spend watching the movies?

## Input

The first line contains three integers $n$, $k$, $c$ ($1 \le n, k, c \le 10^4$) — the number of movies, the number of movies on one page, and the duration of one movie.

## Output

Output a single number $t$, the amount of time that Deadpool will spend watching the movies.

## Examples

| standard input | standard output |
|---|---|
| 20 3 60 | 120 |
| 65 10 30 | 150 |
| 100 20 90 | 1800 |

## Note

In the first example, we have 20 movies and 3 movies on each page except the last one, so there are 2 movies on the last page, and watching those two movies will take a total of 120 minutes.

In the second example, there are a total of 65 movies and 10 movies per page, so there will be 5 movies on the last page, and watching them will take 150 minutes.

In the third example, there are 100 movies, with 20 movies per page, so there will be 20 movies on the last page, and Deadpool will watch them for 1800 minutes.

# Problem B. Catamarans

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

A group of $n$ people plans to go for a ride on catamarans.

As the leader of the group, you have been tasked with ordering the catamarans. You know that one catamaran can hold a weight of no more than 100 kilograms, and you also know the weight of each group member.

You are aware that in your group, a person can weigh either 20, 40, 60, 80, or 100 kilograms.

To spend as little money as possible, you decided to write a program that calculates the minimum number of catamarans needed.

## Input

The first line contains one integer $n$ ($1 \le n \le 1\,000$) — the number of people in the group.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($a_i \in \{20, 40, 60, 80, 100\}$) — the weight of each person.

## Output

Output one integer — the minimum number of catamarans needed.

## Examples

| standard input | standard output |
|---|---|
| 4<br>20 40 80 80 | 3 |
| 4<br>20 40 20 20 | 1 |

## Note

In the first example, we can seat the first two people in one catamaran, the third person in the second catamaran, and the fourth person in the third catamaran. We cannot seat everyone in two catamarans because person 2 cannot sit with person 3 or 4, and person 3 cannot sit with person 4 either.

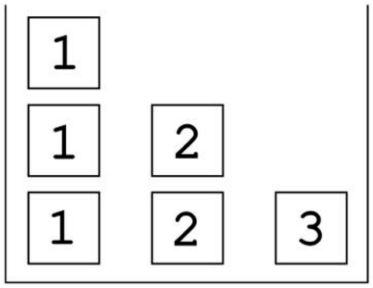
In the second example, we can seat everyone in one catamaran because their total weight is equal to 100 kilograms, which means the catamaran can hold them.

# Problem C. Diagonal Numbers

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

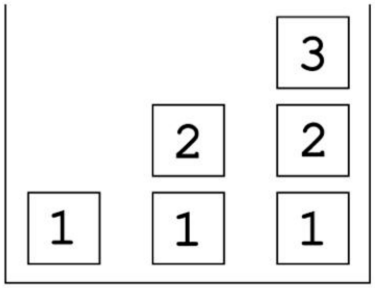
Vasylko was given cubes with numbers from 1 to $n$. One cube had the number $n$, two cubes had the number $n-1$, and so on, with $n-1$ cubes having the number 2, and $n$ cubes having the number 1.

He arranged the cubes in a square frame without a top boundary, so that all cubes were placed below the diagonal that runs from the top left corner to the bottom right corner. In the first column on the left, there are $n$ cubes with the number 1; in the second column, there are $n-1$ cubes with the number 2... in the $n$-th column, there is one cube with the number $n$. That is, in the $i$-th column, there are $n-i+1$ cubes with the number $i$.




Initial state

Vasylko wanted to rearrange them so that all cubes were positioned below the **opposite diagonal**. Thus, in the first row from the bottom, there should be $n$ cubes with the number 1; in the second row, there should be $n-1$ cubes with the number 2... in the $n$-th row, there should be one cube with the number $n$. That is, in the $i$-th row, there should be $n-i+1$ cubes with the number $i$.




Final state

He also decided that he would only move cubes from the top of one column to the top of another (not necessarily adjacent). Help him do this in the most efficient way — find the minimum number of moves he needs and output which moves Vasylko should make.

**You can earn partial points; see details below.**

## Input

The first line contains one integer $n$ ($3 \le n \le 1000$) — the size of the square.

## Output

In the first line, output one integer $k$ ($1 \le k \le 10^6$) — the minimum number of moves needed to rearrange the cubes.

In each of the following $k$ lines, output two integers $a$ and $b$ ($1 \le a, b \le n$; $a \ne b$), where $a$ is the column number from which you take a cube, and $b$ is the column number to which you place the cube.

## Scoring

1. (12 points): $n = 4$;
2. (20 points): $n = 5$;
3. (20 points): $n = 6$;
4. (20 points): $n \le 10$;
5. (20 points): $n \le 100$;
6. (8 points): no additional restrictions.

You can earn half the points for each block if you output the correct $k$ for each test in the block. You will also receive a quarter of the points if you output an incorrect $k$, but the instructions are correct; in this case, $k \le 10^6$.

Note that to earn half the points, you need to output the correct $k$ and

- either output nothing else; that is, output only $k$, but no instructions;
- or output all $k$ instructions completely, where all column numbers are from 1 to $n$. There are no additional requirements for them.

If you output, for example, only a few instructions, or too many instructions, or numbers that are not column numbers, etc., you will receive 0 points.

**Partial points are not awarded for the example from the statement, as it is scored at 0 points.**

The output for half points when $n = 3$ could look like this:

```
8
```

or like this:

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

The output for a quarter of the points when $n = 3$ could look like this:

```
10
1 3
3 1
3 2
1 3
2 1
2 3
1 3
```

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

If you output a number of instructions that does not equal $k$, you will receive 0 points:

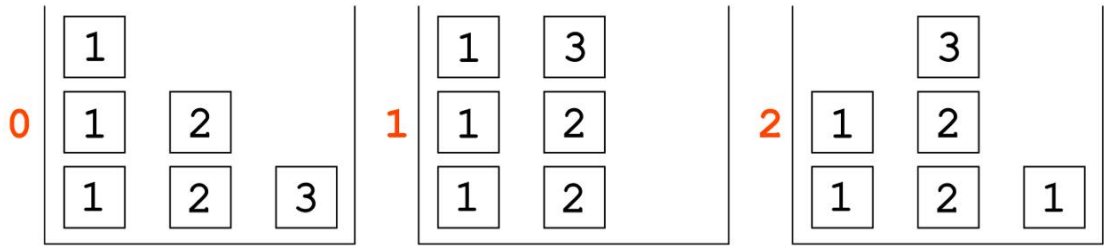
```
8
1 3
3 1
3 2
1 3
2 1
2 3
1 3
2 3
1 2
3 2
```

## Example

| standard input | standard output |
|---|---|
| 3 | 8<br>3 2<br>1 3<br>2 1<br>2 3<br>1 3<br>2 3<br>1 2<br>3 2 |

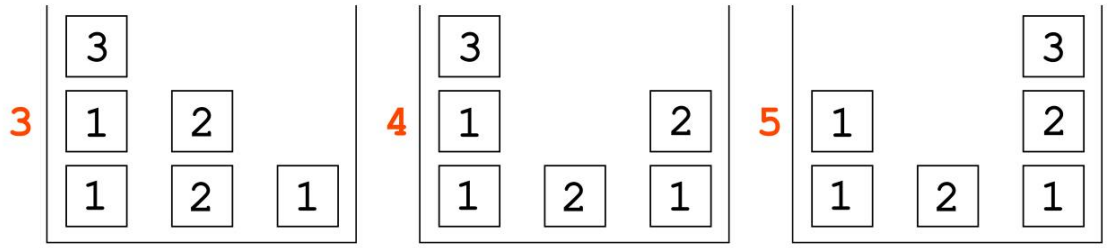
## Note

First, we will place the last column. Therefore, the only first move to rearrange the cubes in the minimum number of moves may be to move 3 from the third column to the second, because otherwise we will not be able to place 1 at the beginning of the third column if we do not move 3 or do not move it to the second column.



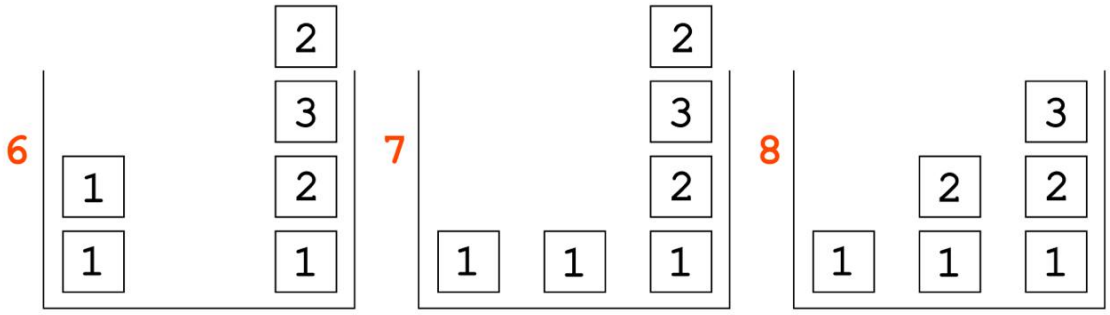

First 3 states

After we have placed 1, we need to place 2 and 3 in their places in the third column, again the only way to do this is to first move 3 to the first column, because otherwise we will not be able to place 2 in the third column in the next move, and then place 3 in its place.




Next 3 states

After this, we want to correctly place the second column, and for 1 to be the first number, we first need to take 2, which we can only take in the last column. After we have placed 1, we only need to return 2 to the second column to achieve the desired state of the cubes.




Final 3 states

# Problem D. Odd Rows

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 1 second |
| Memory limit: | 256 megabytes |

Once, s1mple approached Kostya, a well-known problem solver, and said:

"*If you want to become better, you need constant practice. Here is a problem for training:*"

Given a matrix $a$ of size $n \times m$ ($n$ — number of rows; $m$ — number of columns), where each element is either 0 or 1. It is known that each column contains exactly $c_i$ ones. The elements within each column can be arranged freely. The goal is to maximize the number of rows with an odd number of ones and to find such a matrix.

Kostya silently nodded, sat down at the table, and began to work, knowing that every practice brings him closer to mastery.

Kostya couldn't solve the problem and is asking you to help him solve it.

**You can earn partial points if you only find the number, not the matrix. More details below.**

## Input

The first line contains two integers $n$ and $m$ ($1 \le n \cdot m \le 10^6$) — the dimensions of the matrix.

The second line contains $m$ integers $c_1, c_2, \ldots, c_m$ ($0 \le c_i \le n$) — the number of ones in each column.

## Output

In the first line, output a single integer $t$ ($0 \le t \le n$) — the number of rows in the matrix with an odd sum.

In each of the next $n$ lines, output $m$ integers $a_{ij}$ ($0 \le a_{ij} \le 1$) — the numbers of the matrix.

## Scoring

1. (10 points): $n, m \le 5$;
2. (8 points): the number of ones in the matrix does not exceed $n$;
3. (20 points): the number of ones in each column does not exceed $n/2$;
4. (14 points): $n, m \le 50$;
5. (14 points): $n \le 3\,000$;
6. (14 points): $n \cdot m \le 3 \cdot 10^5$;
7. (20 points): no additional restrictions.

You can earn half the points for each block if you output the correct $t$ for each test in the block.

Note that to earn partial points, you need to output the correct $t$ and

- either output nothing more; that is, output only $t$, but not the matrix;
- or output the complete matrix consisting of 0s and 1s, which does not have to be correct. For example, one that consists entirely of zeros.

If you output, for example, only a few rows, or too many rows, numbers other than 0 and 1, etc., you will receive 0 points.

The output for half points, for the second example, can look like this:

```
2
```

or like this:

```
2
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

but it cannot look like this:

```
2
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

and it cannot look like this:

```
2
10 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

## Examples

| standard input | standard output |
|---|---|
| 8 4<br>6 1 6 1 | 6<br>1 1 1 0<br>1 0 1 1<br>1 0 1 0<br>1 0 1 0<br>1 0 0 0<br>1 0 0 0<br>0 0 1 0<br>0 0 1 0 |
| 4 4<br>3 0 3 0 | 2<br>1 0 1 0<br>1 0 1 0<br>1 0 0 0<br>0 0 1 0 |
| 7 3<br>4 3 2 | 7<br>1 1 1<br>1 0 0<br>1 0 0<br>1 0 0<br>0 1 0<br>0 1 0<br>0 0 1 |

## Note

In the first example, the first and third columns intersect in at least 4 positions, meaning that if there were only these two columns, we would have 4 even rows, but since there are also two columns with 1

one, we can convert two of them to odd, so the optimal answer will be 6 odd rows.

In the second example, we can ignore the second and fourth columns because they have no ones, meaning they will not change the parity of any row, and the first and third intersect in at least 2 rows, meaning at least two rows will be even, so the answer is 2.

In the third example, the answer is 7, because there exists a matrix that has 7 odd rows and satisfies the condition, and there is no matrix that has more than 7.

# Problem E. Three Queries

| | |
|---|---|
| Input file: | `standard input` |
| Output file: | `standard output` |
| Time limit: | 4 seconds |
| Memory limit: | 256 megabytes |

Given an array $a$ of length $n$ and $q$ queries. We also have a binary array $w$ of infinite length, initially all $w_i = 1$.

There are three types of queries:

1. "1 $x$" — toggle the value of $w_x$ (from 1 to 0, and vice versa).
2. "2 $l$ $r$" — count the number of unique numbers in the array $a$ in the range $[l, r]$ for which $w_{a_i} = 1$ and $l \le i \le r$.
3. "3 $x$ $t$" — assign the value $t$ to $a_x$.

Provide an answer for each query of the second type.

## Input

The first line contains two integers $n$ and $q$ ($1 \le n, q \le 3 \cdot 10^5$) — the length of the array and the number of queries.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the values of the elements of the array.

Each of the following $q$ lines starts with an integer *type* ($1 \le type \le 3$) — the type number of each query:

1. If $type = 1$, the query contains one integer $x$ ($1 \le x \le 10^9$) — toggle the value of $w_x$.
2. If $type = 2$, the query contains two integers $l$ and $r$ ($1 \le l \le r \le n$) — count the number of unique numbers in the array $a$ in the range $[l, r]$ for which $w_{a_i} = 1$ and $l \le i \le r$.
3. If $type = 3$, the query contains two integers $x$ and $t$ ($1 \le x \le n, 1 \le t \le 10^9$) — replace the value of $a_x$ with $t$.

## Output

For each query of the second type, output the number of unique numbers in the range on a separate line.

## Scoring

1. (8 points): $n, q \le 10^3$;
2. (6 points): only queries of type 2; $n = q; l_i = 1$; $r_i = i$;
3. (13 points): only queries of type 2;
4. (10 points): only queries of type 1 and 2; all $a_i$ are pairwise distinct;
5. (14 points): only queries of type 1 and 2; all $w_{a_i}$ can change only once;
6. (7 points): only queries of type 1 and 2;
7. (14 points): only queries of type 2 and 3;
8. (8 points): at any moment, $a_i \le 100$;
9. (10 points): $n, q \le 5 \cdot 10^4$;

10. (10 points): without additional constraints.

## Example

| standard input | standard output |
|---|---|
| 10 5<br>3 4 3 4 3 2 3 1 2 1<br>2 2 5<br>1 3<br>2 2 5<br>3 4 5<br>2 2 5 | 2<br>1<br>2 |

## Note

In the example for the first query of the second type, the segment looks like $[4, 3, 4, 3]$, meaning there are numbers 3, 4, and $w_3$, $w_4$ are equal to 1, so the answer is 2. After the next query of type 1, $w_3$ becomes 0, so for the next query, the answer is 1. After the next query, the array will look like $[3, 4, 3, 5, 3, 2, 3, 1, 2, 1]$. In the last query, the segment will look like $[4, 3, 5, 3]$, meaning there are numbers $3, 4, 5$, respectively the answer is 2, because $w_3 = 0$.