ABBYY Cup 2.0 - Hard

A1. Educational Game

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

The Smart Beaver from ABBYY began to develop a new educational game for children. The rules of the game are fairly simple and are described below.

The playing field is a sequence of n non-negative integers a_i numbered from 1 to n. The goal of the game is to make numbers $a_1, a_2, ..., a_k$ (i.e. some prefix of the sequence) equal to zero for some fixed k $(k < n)$, and this should be done in the smallest possible number of moves.

One move is choosing an integer i $(1 \le i \le n)$ such that $a_i > 0$ and an integer t $(t \ge 0)$ such that $i + 2^t \le n$. After the values of i and t have been selected, the value of a_i is decreased by 1, and the value of a_{i+2^t} is increased by 1. For example, let $n = 4$ and $a = (1, 0, 1, 2)$, then it is possible to make move $i = 3$, $t = 0$ and get $a = (1, 0, 0, 3)$ or to make move $i = 1$, $t = 1$ and get $a = (0, 0, 2, 2)$ (the only possible other move is $i = 1$, $t = 0$).

You are given n and the initial sequence a_i. The task is to calculate the minimum number of moves needed to make the first k elements of the original sequence equal to zero for each possible k $(1 \le k < n)$.

Input

The first input line contains a single integer n. The second line contains n integers a_i $(0 \le a_i \le 10^4)$, separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 300$

The input limitations for getting 50 points are:

- $1 \le n \le 2000$

The input limitations for getting 100 points are:

- $1 \le n \le 10^5$

Output

Print exactly $n - 1$ lines: the k-th output line must contain the minimum number of moves needed to make the first k elements of the original sequence a_i equal to zero.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams, or the `%I64d` specifier.

Sample test(s)

input

```
4
1 0 1 2
```

output

```
1
1
3
```

input

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

output

```
1
3
6
10
16
24
40
```

A2. Educational Game

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

The Smart Beaver from ABBYY began to develop a new educational game for children. The rules of the game are fairly simple and are described below.

The playing field is a sequence of n non-negative integers a_i numbered from 1 to n. The goal of the game is to make numbers $a_1, a_2, ..., a_k$ (i.e. some prefix of the sequence) equal to zero for some fixed k $(k < n)$, and this should be done in the smallest possible number of moves.

One move is choosing an integer i $(1 \le i \le n)$ such that $a_i > 0$ and an integer t $(t \ge 0)$ such that $i + 2^t \le n$. After the values of i and t have been selected, the value of a_i is decreased by 1, and the value of a_{i+2^t} is increased by 1. For example, let $n = 4$ and $a = (1, 0, 1, 2)$, then it is possible to make move $i = 3$, $t = 0$ and get $a = (1, 0, 0, 3)$ or to make move $i = 1$, $t = 1$ and get $a = (0, 0, 2, 2)$ (the only possible other move is $i = 1$, $t = 0$).

You are given n and the initial sequence a_i. The task is to calculate the minimum number of moves needed to make the first k elements of the original sequence equal to zero for each possible k $(1 \le k < n)$.

Input

The first input line contains a single integer n. The second line contains n integers a_i $(0 \le a_i \le 10^4)$, separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 300$

The input limitations for getting 50 points are:

- $1 \le n \le 2000$

The input limitations for getting 100 points are:

- $1 \le n \le 10^5$

Output

Print exactly n - 1 lines: the k-th output line must contain the minimum number of moves needed to make the first k elements of the original sequence a_i equal to zero.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams, or the `%I64d` specifier.

Sample test(s)

input

```
4
1 0 1 2
```

output

```
1
1
3
```

input

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

output

```
1
3
6
10
16
24
40
```

A3. Educational Game

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

The Smart Beaver from ABBYY began to develop a new educational game for children. The rules of the game are fairly simple and are described below.

The playing field is a sequence of n non-negative integers a_i numbered from 1 to n. The goal of the game is to make numbers $a_1, a_2, ..., a_k$ (i.e. some prefix of the sequence) equal to zero for some fixed k $(k < n)$, and this should be done in the smallest possible number of moves.

One move is choosing an integer i $(1 \le i \le n)$ such that $a_i > 0$ and an integer t $(t \ge 0)$ such that $i + 2^t \le n$. After the values of i and t have been selected, the value of a_i is decreased by 1, and the value of a_{i+2^t} is increased by 1. For example, let $n = 4$ and $a = (1, 0, 1, 2)$, then it is possible to make move $i = 3$, $t = 0$ and get $a = (1, 0, 0, 3)$ or to make move $i = 1$, $t = 1$ and get $a = (0, 0, 2, 2)$ (the only possible other move is $i = 1$, $t = 0$).

You are given n and the initial sequence a_i. The task is to calculate the minimum number of moves needed to make the first k elements of the original sequence equal to zero for each possible k $(1 \le k < n)$.

Input

The first input line contains a single integer n. The second line contains n integers a_i $(0 \le a_i \le 10^4)$, separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 300$

The input limitations for getting 50 points are:

- $1 \le n \le 2000$

The input limitations for getting 100 points are:

- $1 \le n \le 10^5$

Output

Print exactly n - 1 lines: the k-th output line must contain the minimum number of moves needed to make the first k elements of the original sequence a_i equal to zero.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams, or the `%I64d` specifier.

Sample test(s)

input

```
4
1 0 1 2
```

output

```
1
1
3
```

input

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

output

```
1
3
6
10
16
24
40
```

B1. Greedy Merchants

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

In ABBYY a wonderful Smart Beaver lives. This time, he began to study history. When he read about the Roman Empire, he became interested in the life of merchants.

The Roman Empire consisted of n cities numbered from 1 to n. It also had m bidirectional roads numbered from 1 to m. Each road connected two different cities. Any two cities were connected by no more than one road.

We say that there is a path between cities c_1 and c_2 if there exists a finite sequence of cities $t_1, t_2, ..., t_p$ $(p \geq 1)$ such that:

- $t_1 = c_1$
- $t_p = c_2$
- for any i $(1 \leq i < p)$, cities t_i and t_{i+1} are connected by a road

We know that there existed a path between any two cities in the Roman Empire.

In the Empire k merchants lived numbered from 1 to k. For each merchant we know a pair of numbers s_i and l_i, where s_i is the number of the city where this merchant's warehouse is, and l_i is the number of the city where his shop is. The shop and the warehouse could be located in different cities, so the merchants had to deliver goods from the warehouse to the shop.

Let's call a road *important* for the merchant if its destruction threatens to ruin the merchant, that is, without this road there is no path from the merchant's warehouse to his shop. Merchants in the Roman Empire are very greedy, so each merchant pays a tax (1 dinar) only for those roads which are important for him. In other words, each merchant pays d_i dinars of tax, where d_i ($d_i \geq 0$) is the number of roads important for the i-th merchant.

The tax collection day came in the Empire. The Smart Beaver from ABBYY is very curious by nature, so he decided to count how many dinars each merchant had paid that day. And now he needs your help.

Input

The first input line contains two integers n and m, separated by a space, n is the number of cities, and m is the number of roads in the empire.

The following m lines contain pairs of integers a_i, b_i $(1 \leq a_i, b_i \leq n, a_i \neq b_i)$, separated by a space — the numbers of cities connected by the i-th road. It is guaranteed that any two cities are connected by no more than one road and that there exists a path between any two cities in the Roman Empire.

The next line contains a single integer k — the number of merchants in the empire.

The following k lines contain pairs of integers s_i, l_i $(1 \leq s_i, l_i \leq n)$, separated by a space, — s_i is the number of the city in which the warehouse of the i-th merchant is located, and l_i is the number of the city in which the shop of the i-th merchant is located.

The input limitations for getting 20 points are:

- $1 \leq n \leq 200$
- $1 \leq m \leq 200$
- $1 \leq k \leq 200$

The input limitations for getting 50 points are:

- $1 \leq n \leq 2000$
- $1 \leq m \leq 2000$
- $1 \leq k \leq 2000$

The input limitations for getting 100 points are:

- $1 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq k \leq 10^5$

Output

Print exactly k lines, the i-th line should contain a single integer d_i — the number of dinars that the i-th merchant paid.

Sample test(s)

input

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

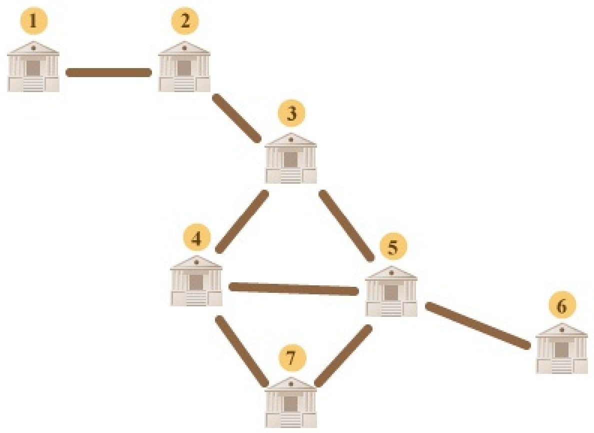
```
4 7
4
1 5
2 4
2 6
4 7
```

output

```
2
1
2
0
```

Note

The given sample is illustrated in the figure below.



Let's describe the result for the first merchant. The merchant's warehouse is located in city 1 and his shop is in city 5. Let us note that if either road, $(1, 2)$ or $(2, 3)$ is destroyed, there won't be any path between cities 1 and 5 anymore. If any other road is destroyed, the path will be preserved. That's why for the given merchant the answer is 2.

B2. Greedy Merchants

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

In ABBYY a wonderful Smart Beaver lives. This time, he began to study history. When he read about the Roman Empire, he became interested in the life of merchants.

The Roman Empire consisted of n cities numbered from 1 to n. It also had m bidirectional roads numbered from 1 to m. Each road connected two different cities. Any two cities were connected by no more than one road.

We say that there is a path between cities c_1 and c_2 if there exists a finite sequence of cities $t_1, t_2, ..., t_p$ $(p \geq 1)$ such that:

- $t_1 = c_1$
- $t_p = c_2$
- for any i $(1 \leq i < p)$, cities t_i and t_{i+1} are connected by a road

We know that there existed a path between any two cities in the Roman Empire.

In the Empire k merchants lived numbered from 1 to k. For each merchant we know a pair of numbers s_i and l_i, where s_i is the number of the city where this merchant's warehouse is, and l_i is the number of the city where his shop is. The shop and the warehouse could be located in different cities, so the merchants had to deliver goods from the warehouse to the shop.

Let's call a road *important* for the merchant if its destruction threatens to ruin the merchant, that is, without this road there is no path from the merchant's warehouse to his shop. Merchants in the Roman Empire are very greedy, so each merchant pays a tax (1 dinar) only for those roads which are important for him. In other words, each merchant pays d_i dinars of tax, where d_i ($d_i \geq 0$) is the number of roads important for the i-th merchant.

The tax collection day came in the Empire. The Smart Beaver from ABBYY is very curious by nature, so he decided to count how many dinars each merchant had paid that day. And now he needs your help.

Input

The first input line contains two integers n and m, separated by a space, n is the number of cities, and m is the number of roads in the empire.

The following m lines contain pairs of integers a_i, b_i $(1 \leq a_i, b_i \leq n, a_i \neq b_i)$, separated by a space — the numbers of cities connected by the i-th road. It is guaranteed that any two cities are connected by no more than one road and that there exists a path between any two cities in the Roman Empire.

The next line contains a single integer k — the number of merchants in the empire.

The following k lines contain pairs of integers s_i, l_i $(1 \leq s_i, l_i \leq n)$, separated by a space, — s_i is the number of the city in which the warehouse of the i-th merchant is located, and l_i is the number of the city in which the shop of the i-th merchant is located.

The input limitations for getting 20 points are:

- $1 \leq n \leq 200$
- $1 \leq m \leq 200$
- $1 \leq k \leq 200$

The input limitations for getting 50 points are:

- $1 \leq n \leq 2000$
- $1 \leq m \leq 2000$
- $1 \leq k \leq 2000$

The input limitations for getting 100 points are:

- $1 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq k \leq 10^5$

Output

Print exactly k lines, the i-th line should contain a single integer d_i — the number of dinars that the i-th merchant paid.

Sample test(s)

input

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

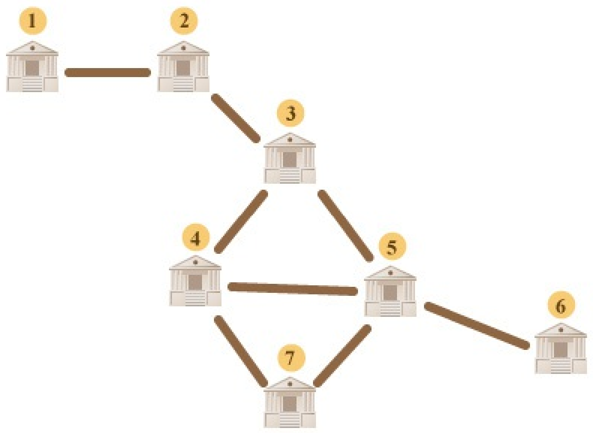
```
4 7
4
1 5
2 4
2 6
4 7
```

output

```
2
1
2
0
```

Note

The given sample is illustrated in the figure below.



Let's describe the result for the first merchant. The merchant's warehouse is located in city 1 and his shop is in city 5. Let us note that if either road, $(1, 2)$ or $(2, 3)$ is destroyed, there won't be any path between cities 1 and 5 anymore. If any other road is destroyed, the path will be preserved. That's why for the given merchant the answer is 2.

B3. Greedy Merchants

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

In ABBYY a wonderful Smart Beaver lives. This time, he began to study history. When he read about the Roman Empire, he became interested in the life of merchants.

The Roman Empire consisted of n cities numbered from 1 to n. It also had m bidirectional roads numbered from 1 to m. Each road connected two different cities. Any two cities were connected by no more than one road.

We say that there is a path between cities c_1 and c_2 if there exists a finite sequence of cities $t_1, t_2, ..., t_p$ $(p \geq 1)$ such that:

- $t_1 = c_1$
- $t_p = c_2$
- for any i $(1 \leq i < p)$, cities t_i and t_{i+1} are connected by a road

We know that there existed a path between any two cities in the Roman Empire.

In the Empire k merchants lived numbered from 1 to k. For each merchant we know a pair of numbers s_i and l_i, where s_i is the number of the city where this merchant's warehouse is, and l_i is the number of the city where his shop is. The shop and the warehouse could be located in different cities, so the merchants had to deliver goods from the warehouse to the shop.

Let's call a road *important* for the merchant if its destruction threatens to ruin the merchant, that is, without this road there is no path from the merchant's warehouse to his shop. Merchants in the Roman Empire are very greedy, so each merchant pays a tax (1 dinar) only for those roads which are important for him. In other words, each merchant pays d_i dinars of tax, where d_i $(d_i \geq 0)$ is the number of roads important for the i-th merchant.

The tax collection day came in the Empire. The Smart Beaver from ABBYY is very curious by nature, so he decided to count how many dinars each merchant had paid that day. And now he needs your help.

Input

The first input line contains two integers n and m, separated by a space, n is the number of cities, and m is the number of roads in the empire.

The following m lines contain pairs of integers a_i, b_i $(1 \leq a_i, b_i \leq n, a_i \neq b_i)$, separated by a space — the numbers of cities connected by the i-th road. It is guaranteed that any two cities are connected by no more than one road and that there exists a path between any two cities in the Roman Empire.

The next line contains a single integer k — the number of merchants in the empire.

The following k lines contain pairs of integers s_i, l_i $(1 \leq s_i, l_i \leq n)$, separated by a space, — s_i is the number of the city in which the warehouse of the i-th merchant is located, and l_i is the number of the city in which the shop of the i-th merchant is located.

The input limitations for getting 20 points are:

- $1 \leq n \leq 200$
- $1 \leq m \leq 200$
- $1 \leq k \leq 200$

The input limitations for getting 50 points are:

- $1 \leq n \leq 2000$
- $1 \leq m \leq 2000$
- $1 \leq k \leq 2000$

The input limitations for getting 100 points are:

- $1 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq k \leq 10^5$

Output

Print exactly k lines, the i-th line should contain a single integer d_i — the number of dinars that the i-th merchant paid.

Sample test(s)

input

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

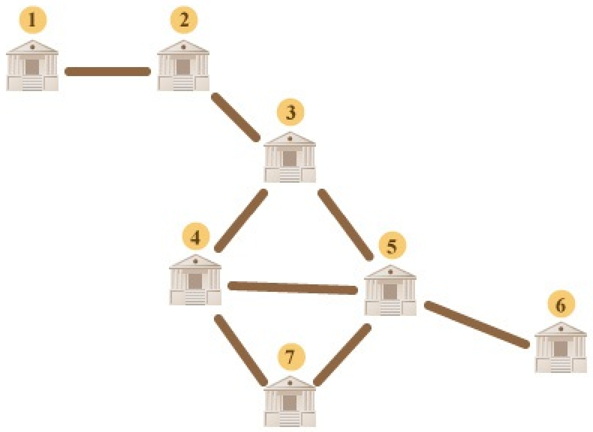
```
4 7
4
1 5
2 4
2 6
4 7
```

output

```
2
1
2
0
```

Note

The given sample is illustrated in the figure below.



Let's describe the result for the first merchant. The merchant's warehouse is located in city 1 and his shop is in city 5. Let us note that if either road, $(1, 2)$ or $(2, 3)$ is destroyed, there won't be any path between cities 1 and 5 anymore. If any other road is destroyed, the path will be preserved. That's why for the given merchant the answer is 2.

C1. Smart Beaver and Resolving Collisions

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

The Smart Beaver from ABBYY has a lot of hobbies. One of them is constructing efficient hash tables. One of the most serious problems in hash tables is resolving collisions. The Beaver is interested in this problem very much and he decided to explore it in detail.

We assume that the hash table consists of h cells numbered from 0 to $h - 1$. Objects are added to and removed from it. Every object has its own unique identifier. In addition, every object has a corresponding hash value — an integer between 0 and $h - 1$, inclusive. When an object is added to the table, if the cell corresponding to the hash value of the object is free, then this object goes there. If the cell is already occupied by another object, there is a collision. When an object is deleted from the table, the cell which it occupied becomes empty.

The Smart Beaver has recently learned about the method of linear probing to resolve collisions. It is as follows. Let's say that the hash value for the added object equals t and cell t of the table is already occupied. Then we try to add this object to cell $(t + m) \ mod \ h$. If it is also occupied, then we try cell $(t + 2 \cdot m) \ mod \ h$, then cell $(t + 3 \cdot m) \ mod \ h$, and so on. Note that in some cases it's possible that the new object can not be added to the table. *It is guaranteed that the input for this problem doesn't contain such situations* .

The operation $a \ mod \ b$ means that we take the remainder of the division of number a by number b.

This technique immediately seemed very inoptimal to the Beaver, and he decided to assess its inefficiency. So, you are given a sequence of operations, each of which is either an addition of an object to the table or a deletion of an object from the table. When adding a new object, a sequence of calls to the table is performed. Calls to occupied cells are called dummy. In other words, if the result of the algorithm described above is the object being added to cell $(t + i \cdot m) \ mod \ h \ (i \ge 0)$, then exactly i dummy calls have been performed.

Your task is to calculate the total number of dummy calls to the table for the given sequence of additions and deletions. When an object is deleted from the table, assume that no dummy calls are performed. The table is empty before performing the operations, that is, initially it doesn't contain any objects.

Input

The first line of input contains three integers h, m and n ($1 \le m < h$), separated by spaces, where h is the size of the hash table, m is the number that is used to resolve collisions, n is the number of operations.

The following n lines contains the descriptions of the operations. Their execution order corresponds to the order in which they appear in the input file. Each operation is described by a single line. The operations are described as follows:

- `"+ id hash"`
  This is the format of the operation that adds an object to the table. The first character is "+" (ASCII 43), followed by a single space, then the object identifier id $(0 \le id \le 10^9)$, then another space, and the hash value of the given object $hash$ $(0 \le hash < h)$. The object identifier and the hash value of this object are integers.
- `"- id"`
  This is the format of the operation that deletes an object from the table. The first character is "-" (ASCII 45), followed by a single space, then the object identifier id $(0 \le id \le 10^9)$. The object identifier is an integer.

It is guaranteed that for all addition operations the value of id is unique. It is also guaranteed that the initial data is correct, that is, it's always possible to add an object to the hash table and there won't be any deletions of nonexisting objects.

The input limitations for getting 20 points are:

- $1 \le h \le 5000$
- $1 \le n \le 5000$

The input limitations for getting 50 points are:

- $1 \le h \le 5 \cdot 10^4$
- $1 \le n \le 5 \cdot 10^4$

The input limitations for getting 100 points are:

- $1 \le h \le 2 \cdot 10^5$
- $1 \le n \le 2 \cdot 10^5$

Output

Print a single number — the total number of dummy calls to the hash table.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams and the `%I64d` specifier.

Sample test(s)

input

```
10 2 7
+ 11 0
+ 22 2
+ 33 6
+ 44 0
+ 55 0
- 22
+ 66 0
```

output

```
7
```

input

```
5 1 6
+ 123 0
+ 234 1
+ 345 2
- 234
+ 456 0
+ 567 0
```

output

```
4
```

C2. Smart Beaver and Resolving Collisions

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

The Smart Beaver from ABBYY has a lot of hobbies. One of them is constructing efficient hash tables. One of the most serious problems in hash tables is resolving collisions. The Beaver is interested in this problem very much and he decided to explore it in detail.

We assume that the hash table consists of h cells numbered from 0 to $h - 1$. Objects are added to and removed from it. Every object has its own unique identifier. In addition, every object has a corresponding hash value — an integer between 0 and $h - 1$, inclusive. When an object is added to the table, if the cell corresponding to the hash value of the object is free, then this object goes there. If the cell is already occupied by another object, there is a collision. When an object is deleted from the table, the cell which it occupied becomes empty.

The Smart Beaver has recently learned about the method of linear probing to resolve collisions. It is as follows. Let's say that the hash value for the added object equals t and cell t of the table is already occupied. Then we try to add this object to cell $(t + m)\ mod\ h$. If it is also occupied, then we try cell $(t + 2 \cdot m)\ mod\ h$, then cell $(t + 3 \cdot m)\ mod\ h$, and so on. Note that in some cases it's possible that the new object can not be added to the table. *It is guaranteed that the input for this problem doesn't contain such situations* .

The operation $a\ mod\ b$ means that we take the remainder of the division of number a by number b.

This technique immediately seemed very inoptimal to the Beaver, and he decided to assess its inefficiency. So, you are given a sequence of operations, each of which is either an addition of an object to the table or a deletion of an object from the table. When adding a new object, a sequence of calls to the table is performed. Calls to occupied cells are called dummy. In other words, if the result of the algorithm described above is the object being added to cell $(t + i \cdot m)\ mod\ h$ $(i \ge 0)$, then exactly i dummy calls have been performed.

Your task is to calculate the total number of dummy calls to the table for the given sequence of additions and deletions. When an object is deleted from the table, assume that no dummy calls are performed. The table is empty before performing the operations, that is, initially it doesn't contain any objects.

Input

The first line of input contains three integers h, m and n $(1 \le m < h)$, separated by spaces, where h is the size of the hash table, m is the number that is used to resolve collisions, n is the number of operations.

The following n lines contains the descriptions of the operations. Their execution order corresponds to the order in which they appear in the input file. Each operation is described by a single line. The operations are described as follows:

- `"+ id hash"`
  This is the format of the operation that adds an object to the table. The first character is "+" (ASCII 43), followed by a single space, then the object identifier id $(0 \le id \le 10^9)$, then another space, and the hash value of the given object $hash$ $(0 \le hash < h)$. The object identifier and the hash value of this object are integers.
- `"- id"`
  This is the format of the operation that deletes an object from the table. The first character is "-" (ASCII 45), followed by a single space, then the object identifier id $(0 \le id \le 10^9)$. The object identifier is an integer.

It is guaranteed that for all addition operations the value of id is unique. It is also guaranteed that the initial data is correct, that is, it's always possible to add an object to the hash table and there won't be any deletions of nonexisting objects.

The input limitations for getting 20 points are:

- $1 \le h \le 5000$
- $1 \le n \le 5000$

The input limitations for getting 50 points are:

- $1 \le h \le 5 \cdot 10^4$
- $1 \le n \le 5 \cdot 10^4$

The input limitations for getting 100 points are:

- $1 \le h \le 2 \cdot 10^5$
- $1 \le n \le 2 \cdot 10^5$

Output

Print a single number — the total number of dummy calls to the hash table.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams and the `%I64d` specifier.

Sample test(s)

input

```
10 2 7
+ 11 0
+ 22 2
+ 33 6
+ 44 0
+ 55 0
- 22
+ 66 0
```

output

```
7
```

input

```
5 1 6
+ 123 0
+ 234 1
+ 345 2
- 234
+ 456 0
+ 567 0
```

output

```
4
```

C3. Smart Beaver and Resolving Collisions

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

The Smart Beaver from ABBYY has a lot of hobbies. One of them is constructing efficient hash tables. One of the most serious problems in hash tables is resolving collisions. The Beaver is interested in this problem very much and he decided to explore it in detail.

We assume that the hash table consists of h cells numbered from 0 to $h - 1$. Objects are added to and removed from it. Every object has its own unique identifier. In addition, every object has a corresponding hash value — an integer between 0 and $h - 1$, inclusive. When an object is added to the table, if the cell corresponding to the hash value of the object is free, then this object goes there. If the cell is already occupied by another object, there is a collision. When an object is deleted from the table, the cell which it occupied becomes empty.

The Smart Beaver has recently learned about the method of linear probing to resolve collisions. It is as follows. Let's say that the hash value for the added object equals t and cell t of the table is already occupied. Then we try to add this object to cell $(t + m)\ mod\ h$. If it is also occupied, then we try cell $(t + 2 \cdot m)\ mod\ h$, then cell $(t + 3 \cdot m)\ mod\ h$, and so on. Note that in some cases it's possible that the new object can not be added to the table. *It is guaranteed that the input for this problem doesn't contain such situations* .

The operation $a\ mod\ b$ means that we take the remainder of the division of number a by number b.

This technique immediately seemed very inoptimal to the Beaver, and he decided to assess its inefficiency. So, you are given a sequence of operations, each of which is either an addition of an object to the table or a deletion of an object from the table. When adding a new object, a sequence of calls to the table is performed. Calls to occupied cells are called dummy. In other words, if the result of the algorithm described above is the object being added to cell $(t + i \cdot m)\ mod\ h$ $(i \geq 0)$, then exactly i dummy calls have been performed.

Your task is to calculate the total number of dummy calls to the table for the given sequence of additions and deletions. When an object is deleted from the table, assume that no dummy calls are performed. The table is empty before performing the operations, that is, initially it doesn't contain any objects.

Input

The first line of input contains three integers h, m and n $(1 \leq m < h)$, separated by spaces, where h is the size of the hash table, m is the number that is used to resolve collisions, n is the number of operations.

The following n lines contains the descriptions of the operations. Their execution order corresponds to the order in which they appear in the input file. Each operation is described by a single line. The operations are described as follows:

- `"+ id hash"`
  This is the format of the operation that adds an object to the table. The first character is "+" (ASCII 43), followed by a single space, then the object identifier id $(0 \leq id \leq 10^9)$, then another space, and the hash value of the given object $hash$ $(0 \leq hash < h)$. The object identifier and the hash value of this object are integers.

- `"- id"`
  This is the format of the operation that deletes an object from the table. The first character is "-" (ASCII 45), followed by a single space, then the object identifier id $(0 \leq id \leq 10^9)$. The object identifier is an integer.

It is guaranteed that for all addition operations the value of id is unique. It is also guaranteed that the initial data is correct, that is, it's always possible to add an object to the hash table and there won't be any deletions of nonexisting objects.

The input limitations for getting 20 points are:

- $1 \leq h \leq 5000$
- $1 \leq n \leq 5000$

The input limitations for getting 50 points are:

- $1 \leq h \leq 5 \cdot 10^4$
- $1 \leq n \leq 5 \cdot 10^4$

The input limitations for getting 100 points are:

- $1 \leq h \leq 2 \cdot 10^5$
- $1 \leq n \leq 2 \cdot 10^5$

Output

Print a single number — the total number of dummy calls to the hash table.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout` streams and the `%I64d` specifier.

Sample test(s)

input

```
10 2 7
+ 11 0
+ 22 2
+ 33 6
+ 44 0
+ 55 0
- 22
+ 66 0
```

output

```
7
```

input

```
5 1 6
+ 123 0
+ 234 1
+ 345 2
- 234
+ 456 0
+ 567 0
```

output

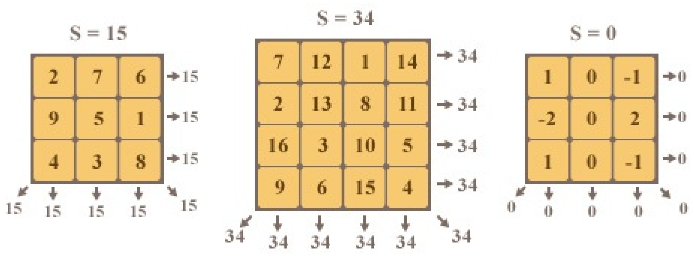
```
4
```

D1. Magic Squares

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

The Smart Beaver from ABBYY loves puzzles. One of his favorite puzzles is the magic square. He has recently had an idea to automate the solution of this puzzle. The Beaver decided to offer this challenge to the ABBYY Cup contestants.

The magic square is a matrix of size $n \times n$. The elements of this matrix are integers. The sum of numbers in each row of the matrix is equal to some number s. The sum of numbers in each column of the matrix is also equal to s. In addition, the sum of the elements on the main diagonal is equal to s and the sum of elements on the secondary diagonal is equal to s. Examples of magic squares are given in the following figure:



Magic squares

You are given a set of n^2 integers a_i. It is required to place these numbers into a square matrix of size $n \times n$ so that they form a magic square. Note that each number must occur in the matrix exactly the same number of times as it occurs in the original set.

It is guaranteed that a solution exists!

Input

The first input line contains a single integer n. The next line contains n^2 integers a_i ($-10^8 \le a_i \le 10^8$), separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 3$

The input limitations for getting 50 points are:

- $1 \le n \le 4$
- It is guaranteed that there are no more than 9 distinct numbers among a_i.

The input limitations for getting 100 points are:

- $1 \le n \le 4$

Output

The first line of the output should contain a single integer s. In each of the following n lines print n integers, separated by spaces and describing the resulting magic square. In the resulting magic square the sums in the rows, columns and diagonals must be equal to s. If there are multiple solutions, you are allowed to print any of them.

Sample test(s)

input

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

output

```
15
2 7 6
9 5 1
4 3 8
```

input

```
3
1 0 -1 0 2 -1 -2 0 1
```

output

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

input

```
2
5 5 5 5
```

output

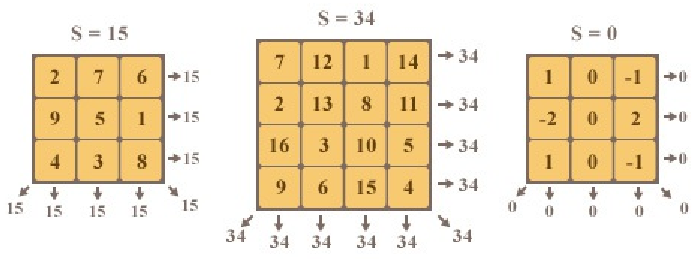
```
10
5 5
5 5
```

D2. Magic Squares

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

The Smart Beaver from ABBYY loves puzzles. One of his favorite puzzles is the magic square. He has recently had an idea to automate the solution of this puzzle. The Beaver decided to offer this challenge to the ABBYY Cup contestants.

The magic square is a matrix of size $n \times n$. The elements of this matrix are integers. The sum of numbers in each row of the matrix is equal to some number s. The sum of numbers in each column of the matrix is also equal to s. In addition, the sum of the elements on the main diagonal is equal to s and the sum of elements on the secondary diagonal is equal to s. Examples of magic squares are given in the following figure:



Magic squares

You are given a set of n^2 integers a_i. It is required to place these numbers into a square matrix of size $n \times n$ so that they form a magic square. Note that each number must occur in the matrix exactly the same number of times as it occurs in the original set.

It is guaranteed that a solution exists!

Input

The first input line contains a single integer n. The next line contains n^2 integers a_i ($-10^8 \le a_i \le 10^8$), separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 3$

The input limitations for getting 50 points are:

- $1 \le n \le 4$
- It is guaranteed that there are no more than 9 distinct numbers among a_i.

The input limitations for getting 100 points are:

- $1 \le n \le 4$

Output

The first line of the output should contain a single integer s. In each of the following n lines print n integers, separated by spaces and describing the resulting magic square. In the resulting magic square the sums in the rows, columns and diagonals must be equal to s. If there are multiple solutions, you are allowed to print any of them.

Sample test(s)

input

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

output

```
15
2 7 6
9 5 1
4 3 8
```

input

```
3
1 0 -1 0 2 -1 -2 0 1
```

output

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

input

```
2
5 5 5 5
```

output

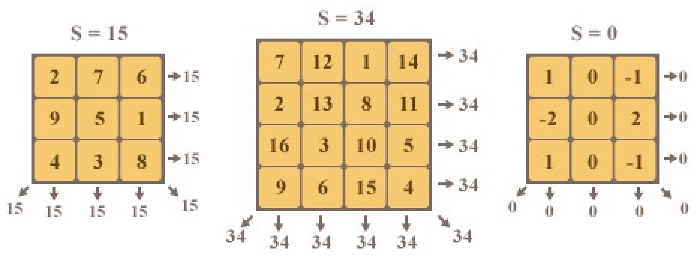
```
10
5 5
5 5
```

D3. Magic Squares

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

The Smart Beaver from ABBYY loves puzzles. One of his favorite puzzles is the magic square. He has recently had an idea to automate the solution of this puzzle. The Beaver decided to offer this challenge to the ABBYY Cup contestants.

The magic square is a matrix of size $n \times n$. The elements of this matrix are integers. The sum of numbers in each row of the matrix is equal to some number s. The sum of numbers in each column of the matrix is also equal to s. In addition, the sum of the elements on the main diagonal is equal to s and the sum of elements on the secondary diagonal is equal to s. Examples of magic squares are given in the following figure:



Magic squares

You are given a set of n^2 integers a_i. It is required to place these numbers into a square matrix of size $n \times n$ so that they form a magic square. Note that each number must occur in the matrix exactly the same number of times as it occurs in the original set.

It is guaranteed that a solution exists!

Input

The first input line contains a single integer n. The next line contains n^2 integers a_i ($-10^8 \le a_i \le 10^8$), separated by single spaces.

The input limitations for getting 20 points are:

- $1 \le n \le 3$

The input limitations for getting 50 points are:

- $1 \le n \le 4$
- It is guaranteed that there are no more than 9 distinct numbers among a_i.

The input limitations for getting 100 points are:

- $1 \le n \le 4$

Output

The first line of the output should contain a single integer s. In each of the following n lines print n integers, separated by spaces and describing the resulting magic square. In the resulting magic square the sums in the rows, columns and diagonals must be equal to s. If there are multiple solutions, you are allowed to print any of them.

Sample test(s)

input

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

output

```
15
2 7 6
9 5 1
4 3 8
```

input

```
3
1 0 -1 0 2 -1 -2 0 1
```

output

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

input

```
2
5 5 5 5
```

output

```
10
5 5
5 5
```

E1. The Beaver's Problem - 2

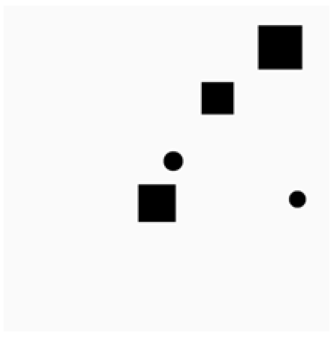
time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Offering the ABBYY Cup participants a problem written by the Smart Beaver is becoming a tradition. He proposed the following problem.

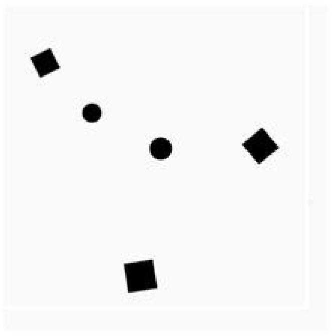
You are given a monochrome image, that is, an image that is composed of two colors (black and white). The image is given in raster form, that is, as a matrix of pixels' colors, and the matrix's size coincides with the size of the image.

The white color on the given image corresponds to the background. Also, the image contains several black geometric shapes. It is known that the image can contain only two types of shapes: squares and circles. Your task is to count the number of circles and the number of squares which the given image contains.

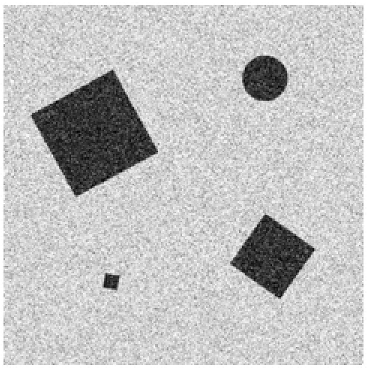
The squares on the image can be rotated arbitrarily. In addition, the image can possibly contain some noise arranged as follows: each pixel of the original image can change its color to the opposite with the probability of 20%.



An example of an image that has no noise and the sides of the squares are parallel to the coordinate axes (two circles and three squares).



An example of an image that has no noise and the squares are rotated arbitrarily (two circles and three squares).



An example of an image that has noise and the squares are rotated arbitrarily (one circle and three squares).

Input

The first input line contains a single integer n $(1000 \le n \le 2000)$, which is the length and the width of the original image.

Next n lines describe the matrix of colors of the image pixels. The i-th line contains exactly n integers a_{ij} $(0 \le a_{ij} \le 1)$, separated by spaces. Value of $a_{ij} = 0$ corresponds to a white pixel and $a_{ij} = 1$ corresponds to a black one.

It is guaranteed that the lengths of the sides of the squares and the diameters of the circles in the image are at least 15 pixels, and the distance between any two figures is at least 10 pixels. It is also guaranteed that a human can easily calculate the number of circles and squares in the original image. The total number of figures in the image doesn't exceed 50.

The input limitations for getting 20 points are:

- These test cases have no noise and the sides of the squares are parallel to the coordinate axes.

The input limitations for getting 50 points are:

- These test cases have no noise, but the squares are rotated arbitrarily.

The input limitations for getting 100 points are:

- These test cases have noise and the squares are rotated arbitrarily.

Output

Print exactly two integers, separated by a single space — the number of circles and the number of squares in the given image, correspondingly.

Sample test(s)

Note

You are given a sample of original data for each difficulty level. The samples are available at `http://codeforces.ru/static/materials/contests/178/e-samples.zip`.

E2. The Beaver's Problem - 2

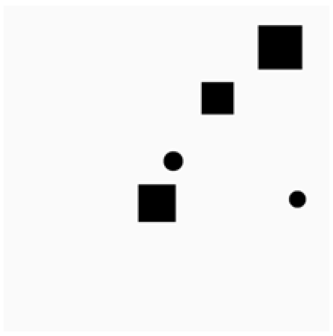
time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Offering the ABBYY Cup participants a problem written by the Smart Beaver is becoming a tradition. He proposed the following problem.

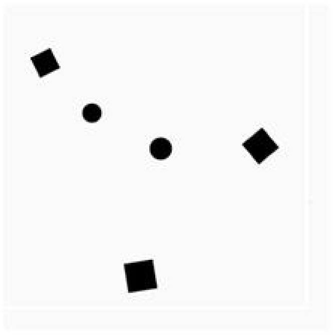
You are given a monochrome image, that is, an image that is composed of two colors (black and white). The image is given in raster form, that is, as a matrix of pixels' colors, and the matrix's size coincides with the size of the image.

The white color on the given image corresponds to the background. Also, the image contains several black geometric shapes. It is known that the image can contain only two types of shapes: squares and circles. Your task is to count the number of circles and the number of squares which the given image contains.

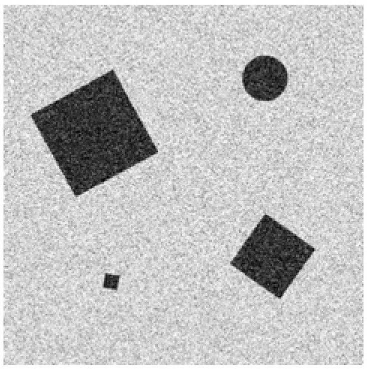
The squares on the image can be rotated arbitrarily. In addition, the image can possibly contain some noise arranged as follows: each pixel of the original image can change its color to the opposite with the probability of 20%.



An example of an image that has no noise and the sides of the squares are parallel to the coordinate axes (two circles and three squares).



An example of an image that has no noise and the squares are rotated arbitrarily (two circles and three squares).



An example of an image that has noise and the squares are rotated arbitrarily (one circle and three squares).

Input

The first input line contains a single integer n ($1000 \le n \le 2000$), which is the length and the width of the original image.

Next n lines describe the matrix of colors of the image pixels. The i-th line contains exactly n integers a_{ij} ($0 \le a_{ij} \le 1$), separated by spaces. Value of $a_{ij} = 0$ corresponds to a white pixel and $a_{ij} = 1$ corresponds to a black one.

It is guaranteed that the lengths of the sides of the squares and the diameters of the circles in the image are at least 15 pixels, and the distance between any two figures is at least 10 pixels. It is also guaranteed that a human can easily calculate the number of circles and squares in the original image. The total number of figures in the image doesn't exceed 50.

The input limitations for getting 20 points are:

- These test cases have no noise and the sides of the squares are parallel to the coordinate axes.

The input limitations for getting 50 points are:

- These test cases have no noise, but the squares are rotated arbitrarily.

The input limitations for getting 100 points are:

- These test cases have noise and the squares are rotated arbitrarily.

Output

Print exactly two integers, separated by a single space — the number of circles and the number of squares in the given image, correspondingly.

Sample test(s)

Note

You are given a sample of original data for each difficulty level. The samples are available at `http://codeforces.ru/static/materials/contests/178/e-samples.zip`.

E3. The Beaver's Problem - 2

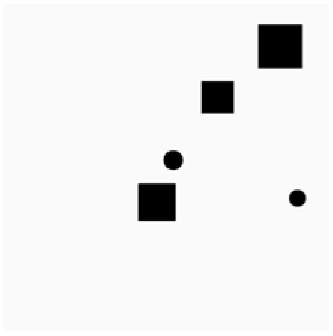
time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Offering the ABBYY Cup participants a problem written by the Smart Beaver is becoming a tradition. He proposed the following problem.

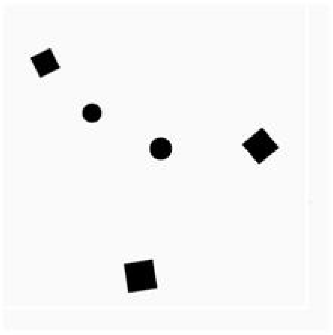
You are given a monochrome image, that is, an image that is composed of two colors (black and white). The image is given in raster form, that is, as a matrix of pixels' colors, and the matrix's size coincides with the size of the image.

The white color on the given image corresponds to the background. Also, the image contains several black geometric shapes. It is known that the image can contain only two types of shapes: squares and circles. Your task is to count the number of circles and the number of squares which the given image contains.

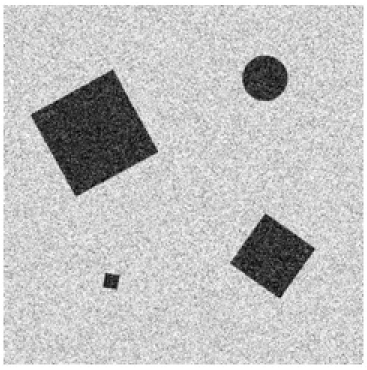
The squares on the image can be rotated arbitrarily. In addition, the image can possibly contain some noise arranged as follows: each pixel of the original image can change its color to the opposite with the probability of 20%.



An example of an image that has no noise and the sides of the squares are parallel to the coordinate axes (two circles and three squares).



An example of an image that has no noise and the squares are rotated arbitrarily (two circles and three squares).



An example of an image that has noise and the squares are rotated arbitrarily (one circle and three squares).

Input

The first input line contains a single integer n ($1000 \le n \le 2000$), which is the length and the width of the original image.

Next n lines describe the matrix of colors of the image pixels. The i-th line contains exactly n integers a_{ij} ($0 \le a_{ij} \le 1$), separated by spaces. Value of $a_{ij} = 0$ corresponds to a white pixel and $a_{ij} = 1$ corresponds to a black one.

It is guaranteed that the lengths of the sides of the squares and the diameters of the circles in the image are at least 15 pixels, and the distance between any two figures is at least 10 pixels. It is also guaranteed that a human can easily calculate the number of circles and squares in the original image. The total number of figures in the image doesn't exceed 50.

The input limitations for getting 20 points are:

- These test cases have no noise and the sides of the squares are parallel to the coordinate axes.

The input limitations for getting 50 points are:

- These test cases have no noise, but the squares are rotated arbitrarily.

The input limitations for getting 100 points are:

- These test cases have noise and the squares are rotated arbitrarily.

Output

Print exactly two integers, separated by a single space — the number of circles and the number of squares in the given image, correspondingly.

Sample test(s)

Note

You are given a sample of original data for each difficulty level. The samples are available at `http://codeforces.ru/static/materials/contests/178/e-samples.zip`.

F1. Representative Sampling

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

The Smart Beaver from ABBYY has a long history of cooperating with the "Institute of Cytology and Genetics". Recently, the Institute staff challenged the Beaver with a new problem. The problem is as follows.

There is a collection of n proteins (not necessarily distinct). Each protein is a string consisting of lowercase Latin letters. The problem that the scientists offered to the Beaver is to select a subcollection of size k from the initial collection of proteins so that the representativity of the selected subset of proteins is maximum possible.

The Smart Beaver from ABBYY did some research and came to the conclusion that the representativity of a collection of proteins can be evaluated by a single number, which is simply calculated. Let's suppose we have a collection $\{a_1, ..., a_k\}$ consisting of k strings describing proteins. The representativity of this collection is the following value:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} f(a_i, a_j),$$

where $f(x, y)$ is the length of the longest common prefix of strings x and y; for example, $f(\text{"abc"}, \text{"abd"}) = 2$, and $f(\text{"ab"}, \text{"bcd"}) = 0$.

Thus, the representativity of collection of proteins $\{\text{"abc"}, \text{"abd"}, \text{"abe"}\}$ equals 6, and the representativity of collection $\{\text{"aaa"}, \text{"ba"}, \text{"ba"}\}$ equals 2.

Having discovered that, the Smart Beaver from ABBYY asked the Cup contestants to write a program that selects, from the given collection of proteins, a subcollection of size k which has the largest possible value of representativity. Help him to solve this problem!

Input

The first input line contains two integers n and k ($1 \le k \le n$), separated by a single space. The following n lines contain the descriptions of proteins, one per line. Each protein is a non-empty string of no more than 500 characters consisting of only lowercase Latin letters (a...z). Some of the strings may be equal.

The input limitations for getting 20 points are:

- $1 \le n \le 20$

The input limitations for getting 50 points are:

- $1 \le n \le 100$

The input limitations for getting 100 points are:

- $1 \le n \le 2000$

Output

Print a single number denoting the largest possible value of representativity that a subcollection of size k of the given collection of proteins can have.

Sample test(s)

input
```
3 2
aba
bzd
abq
```
output
```
2
```

input
```
4 3
eee
rrr
ttt
qqq
```
output
```
0
```

input
```
4 3
aaa
abba
abbc
abbd
```
output
```
9
```

F2. Representative Sampling

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

The Smart Beaver from ABBYY has a long history of cooperating with the "Institute of Cytology and Genetics". Recently, the Institute staff challenged the Beaver with a new problem. The problem is as follows.

There is a collection of n proteins (not necessarily distinct). Each protein is a string consisting of lowercase Latin letters. The problem that the scientists offered to the Beaver is to select a subcollection of size k from the initial collection of proteins so that the representativity of the selected subset of proteins is maximum possible.

The Smart Beaver from ABBYY did some research and came to the conclusion that the representativity of a collection of proteins can be evaluated by a single number, which is simply calculated. Let's suppose we have a collection $\{a_1, ..., a_k\}$ consisting of k strings describing proteins. The representativity of this collection is the following value:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} f(a_i, a_j),$$

where $f(x, y)$ is the length of the longest common prefix of strings x and y; for example, $f($"abc", "abd"$) = 2$, and $f($"ab", "bcd"$) = 0$.

Thus, the representativity of collection of proteins $\{$"abc", "abd", "abe"$\}$ equals 6, and the representativity of collection $\{$"aaa", "ba", "ba"$\}$ equals 2.

Having discovered that, the Smart Beaver from ABBYY asked the Cup contestants to write a program that selects, from the given collection of proteins, a subcollection of size k which has the largest possible value of representativity. Help him to solve this problem!

Input

The first input line contains two integers n and k ($1 \le k \le n$), separated by a single space. The following n lines contain the descriptions of proteins, one per line. Each protein is a non-empty string of no more than 500 characters consisting of only lowercase Latin letters (a...z). Some of the strings may be equal.

The input limitations for getting 20 points are:

- $1 \le n \le 20$

The input limitations for getting 50 points are:

- $1 \le n \le 100$

The input limitations for getting 100 points are:

- $1 \le n \le 2000$

Output

Print a single number denoting the largest possible value of representativity that a subcollection of size k of the given collection of proteins can have.

Sample test(s)

input
```
3 2
aba
bzd
abq
```
output
```
2
```

input
```
4 3
eee
rrr
ttt
qqq
```
output
```
0
```

input
```
4 3
aaa
abba
abbc
abbd
```
output
```
9
```

F3. Representative Sampling

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

The Smart Beaver from ABBYY has a long history of cooperating with the "Institute of Cytology and Genetics". Recently, the Institute staff challenged the Beaver with a new problem. The problem is as follows.

There is a collection of n proteins (not necessarily distinct). Each protein is a string consisting of lowercase Latin letters. The problem that the scientists offered to the Beaver is to select a subcollection of size k from the initial collection of proteins so that the representativity of the selected subset of proteins is maximum possible.

The Smart Beaver from ABBYY did some research and came to the conclusion that the representativity of a collection of proteins can be evaluated by a single number, which is simply calculated. Let's suppose we have a collection $\{a_1, ..., a_k\}$ consisting of k strings describing proteins. The representativity of this collection is the following value:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} f(a_i, a_j),$$

where $f(x, y)$ is the length of the longest common prefix of strings x and y; for example, $f(\texttt{"abc"}, \texttt{"abd"}) = 2$, and $f(\texttt{"ab"}, \texttt{"bcd"}) = 0$.

Thus, the representativity of collection of proteins $\{\texttt{"abc"}, \texttt{"abd"}, \texttt{"abe"}\}$ equals 6, and the representativity of collection $\{\texttt{"aaa"}, \texttt{"ba"}, \texttt{"ba"}\}$ equals 2.

Having discovered that, the Smart Beaver from ABBYY asked the Cup contestants to write a program that selects, from the given collection of proteins, a subcollection of size k which has the largest possible value of representativity. Help him to solve this problem!

Input

The first input line contains two integers n and k ($1 \le k \le n$), separated by a single space. The following n lines contain the descriptions of proteins, one per line. Each protein is a non-empty string of no more than 500 characters consisting of only lowercase Latin letters ($\texttt{a}...\texttt{z}$). Some of the strings may be equal.

The input limitations for getting 20 points are:

- $1 \le n \le 20$

The input limitations for getting 50 points are:

- $1 \le n \le 100$

The input limitations for getting 100 points are:

- $1 \le n \le 2000$

Output

Print a single number denoting the largest possible value of representativity that a subcollection of size k of the given collection of proteins can have.

Sample test(s)

input

```
3 2
aba
bzd
abq
```

output

```
2
```

input

```
4 3
eee
rrr
ttt
qqq
```

output

```
0
```

input

```
4 3
aaa
abba
abbc
abbd
```

output

```
9
```

Codeforces (c) Copyright 2010-2015 Mike Mirzayanov
The only programming contests Web 2.0 platform