15-295 Fall 2018 #6

A. Palindrome Pairs

2 seconds, 256 megabytes

After learning a lot about space exploration, a little girl named Ana wants to change the subject.

Ana is a girl who loves palindromes (string that can be read the same backwards as forward). She has learned how to check for a given string whether it's a palindrome or not, but soon she grew tired of this problem, so she came up with a more interesting one and she needs your help to solve it:

You are given an array of strings which consist of only small letters of the alphabet. Your task is to find **how many** palindrome pairs are there in the array. A palindrome pair is a pair of strings such that the following condition holds: **at least one** permutation of the concatenation of the two strings is a palindrome. In other words, if you have two strings, let's say "aab" and "abcac", and you concatenate them into "aababcac", we have to check if there exists a permutation of this new string such that it is a palindrome (in this case there exists the permutation "aabccbaa").

Two pairs are considered different if the strings are located on **different indices**. The pair of strings with indices (i, j) is considered **the same** as the pair (j, i).

Input

The first line contains a positive integer N ($1 \le N \le 100\,000$), representing the length of the input array.

Eacg of the next N lines contains a string (consisting of lowercase English letters from 'a' to 'z') — an element of the input array.

The total number of characters in the input array will be less than $1\,000\,000$.

Output

Output one number, representing **how many palindrome pairs** there are in the array.

input
```
3
aa
bb
cd
```
output
```
1
```

input
```
6
aab
abcac
dffe
ed
aa
aade
```
output
```
6
```

The first example:

1. $\mathtt{aa} + \mathtt{bb} \to \mathtt{abba}$.

The second example:

1. $\mathtt{aab} + \mathtt{abcac} = \mathtt{aababcac} \to \mathtt{aabccbaa}$
2. $\mathtt{aab} + \mathtt{aa} = \mathtt{aabaa}$
3. $\mathtt{abcac} + \mathtt{aa} = \mathtt{abcacaa} \to \mathtt{aacbcaa}$
4. $\mathtt{dffe} + \mathtt{ed} = \mathtt{dffeed} \to \mathtt{fdeedf}$
5. $\mathtt{dffe} + \mathtt{aade} = \mathtt{dffeaade} \to \mathtt{adfaafde}$
6. $\mathtt{ed} + \mathtt{aade} = \mathtt{edaade} \to \mathtt{aeddea}$

B. Bicolorings

2 seconds, 256 megabytes

You are given a grid, consisting of 2 rows and n columns. Each cell of this grid should be colored either black or white.

Two cells are considered neighbours if they have a **common border** and share the same color. Two cells A and B belong to the same component if they are neighbours, or if there is a neighbour of A that belongs to the same component with B.

Let's call some bicoloring *beautiful* if it has exactly k components.

Count the number of *beautiful* bicolorings. The number can be big enough, so print the answer modulo 998244353.

Input

The only line contains two integers n and k ($1 \le n \le 1000$, $1 \le k \le 2n$) — the number of columns in a grid and the number of components required.

Output

Print a single integer — the number of *beautiful* bicolorings modulo 998244353.

input
```
3 4
```
output
```
12
```

input
```
4 1
```
output
```
2
```

input
```
1 2
```
output
```
2
```

One of possible bicolorings in sample 1:

C. Party Lemonade

1 second, 256 megabytes

A New Year party is not a New Year party without lemonade! As usual, you are expecting a lot of guests, and buying lemonade has already become a pleasant necessity.

Your favorite store sells lemonade in bottles of n different volumes at different costs. A single bottle of type i has volume 2^{i-1} liters and costs c_i roubles. The number of bottles of each type in the store can be considered infinite.

You want to buy at least L liters of lemonade. How many roubles do you have to spend?

Input

The first line contains two integers n and L ($1 \le n \le 30$; $1 \le L \le 10^9$) — the number of types of bottles in the store and the required amount of lemonade in liters, respectively.

The second line contains n integers $c_1, c_2, ..., c_n$ ($1 \le c_i \le 10^9$) — the costs of bottles of different types.

Output

Output a single integer — the smallest number of roubles you have to pay in order to buy at least L liters of lemonade.

input

```
4 12
20 30 70 90
```

output

```
150
```

input

```
4 3
10000 1000 100 10
```

output

```
10
```

input

```
4 3
10 100 1000 10000
```

output

```
30
```

input

```
5 787787787
123456789 234567890 345678901 456789012 987654321
```

output

```
44981600785557577
```

In the first example you should buy one 8-liter bottle for 90 roubles and two 2-liter bottles for 30 roubles each. In total you'll get 12 liters of lemonade for just 150 roubles.

In the second example, even though you need only 3 liters, it's cheaper to buy a single 8-liter bottle for 10 roubles.

In the third example it's best to buy three 1-liter bottles for 10 roubles each, getting three liters for 30 roubles.

D. Careful Maneuvering

2 seconds, 256 megabytes

There are two small spaceship, surrounded by two groups of enemy larger spaceships. The space is a two-dimensional plane, and one group of the enemy spaceships is positioned in such a way that they all have integer y-coordinates, and their x-coordinate is equal to -100, while the second group is positioned in such a way that they all have integer y -coordinates, and their x-coordinate is equal to 100.

Each spaceship in both groups will simultaneously shoot two laser shots (infinite ray that destroys any spaceship it touches), one towards each of the small spaceships, all at the same time. The small spaceships will be able to avoid all the laser shots, and now want to position themselves at some locations with $x = 0$ (with not necessarily integer y-coordinates), such that the rays shot at them would destroy as many of the enemy spaceships as possible. Find the largest numbers of spaceships that can be destroyed this way, assuming that the enemy spaceships can't avoid laser shots.

Input

The first line contains two integers n and m ($1 \le n, m \le 60$), the number of enemy spaceships with $x = -100$ and the number of enemy spaceships with $x = 100$, respectively.

The second line contains n integers $y_{1,1}, y_{1,2}, \ldots, y_{1,n}$ ($|y_{1,i}| \le 10\,000$) — the y-coordinates of the spaceships in the first group.

The third line contains m integers $y_{2,1}, y_{2,2}, \ldots, y_{2,m}$ ($|y_{2,i}| \le 10\,000$) — the y-coordinates of the spaceships in the second group.

The y coordinates are not guaranteed to be unique, even within a group.

Output

Print a single integer – the largest number of enemy spaceships that can be destroyed.

input

```
3 9
1 2 3
1 2 3 7 8 9 11 12 13
```

output

```
9
```

input

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

output

```
10
```

In the first example the first spaceship can be positioned at $(0, 2)$, and the second – at $(0, 7)$. This way all the enemy spaceships in the first group and 6 out of 9 spaceships in the second group will be destroyed.

In the second example the first spaceship can be positioned at $(0, 3)$, and the second can be positioned anywhere, it will be sufficient to destroy all the enemy spaceships.

E. Random Task

1 second, 256 megabytes

One day, after a difficult lecture a diligent student Sasha saw a graffitied desk in the classroom. She came closer and read: "Find such positive integer n, that among numbers $n+1$, $n+2$, ..., $2 \cdot n$ there are exactly m numbers which binary representation contains exactly k digits one".

The girl got interested in the task and she asked you to help her solve it. Sasha knows that you are afraid of large numbers, so she guaranteed that there is an answer that doesn't exceed 10^{18}.

Input

The first line contains two space-separated integers, m and k ($0 \le m \le 10^{18}$; $1 \le k \le 64$).

Output

Print the required number n ($1 \le n \le 10^{18}$). If there are multiple answers, print any of them.

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

input
```
3 2
```
output
```
5
```

F. The Shortest Statement

4 seconds, 256 megabytes

You are given a weighed undirected **connected** graph, consisting of n vertices and m edges.

You should answer q queries, the i-th query is to find the shortest distance between vertices u_i and v_i.

Input

The first line contains two integers n and m $(1 \leq n, m \leq 10^5, m - n \leq 20)$ — the number of vertices and edges in the graph.

Next m lines contain the edges: the i-th edge is a triple of integers v_i, u_i, d_i $(1 \leq u_i, v_i \leq n, 1 \leq d_i \leq 10^9, u_i \neq v_i)$. This triple means that there is an edge between vertices u_i and v_i of weight d_i. It is guaranteed that graph contains no self-loops and multiple edges.

The next line contains a single integer q $(1 \leq q \leq 10^5)$ — the number of queries.

Each of the next q lines contains two integers u_i and v_i $(1 \leq u_i, v_i \leq n)$ — descriptions of the queries.

Pay attention to the restriction $m - n \leq 20$.

Output

Print q lines.

The i-th line should contain the answer to the i-th query — the shortest distance between vertices u_i and v_i.

input
```
3 3
1 2 3
2 3 1
3 1 5
3
1 2
1 3
2 3
```
output
```
3
4
1
```

input
```
8 13
1 2 4
2 3 6
3 4 1
4 5 12
5 6 3
6 7 8
7 8 7
1 4 1
1 8 3
2 6 9
2 7 1
4 6 3
6 8 2
8
1 5
1 7
2 3
2 8
3 7
3 4
6 8
7 8
```
output
```
7
5
6
7
7
1
2
7
```