Problem A - Prime abundance
UAC3 - 2015

Jack is an inquisitive little boy. When learning about prime numbers in school he wanted to try to figure out which numbers have a lot prime factors. A prime number is a positive integer whose only positive divisors are 1 and itself. The first few prime numbers are:

$$2, 3, 5, 7, 11, 13, 17, 19, \ldots$$

Each integer can be expressed as a product of primes. For instance $12 = 2\times2\times3$ has three prime factors, 7 has only one, itself, and 1 has none.

He just learned all the numbers up to n and asked his teacher which of those numbers has the most prime factors. However his teacher did not know the answer so he turned to you for help.

Example: If $n = 6$ we have:

n	prime factors	number of prime factors
1	-	0
2	2	1
3	3	1
4	2×2	2
5	5	1
6	2×3	2

The answer is 2 because the integer less than or equal to 6 that has the most prime factors has 2 prime factors.

Input

A single integer n as described above.

Constraints

- $1 \leq n \leq 2^{31} - 1$

Output

The number of prime factors of the number less than or equal to n that has the most prime factors.

Sample Test Cases

Sample Input 1

```
555
```

Sample Output 1

```
9
```

Sample Input 2

```
123456789
```

Sample Output 2

```
26
```

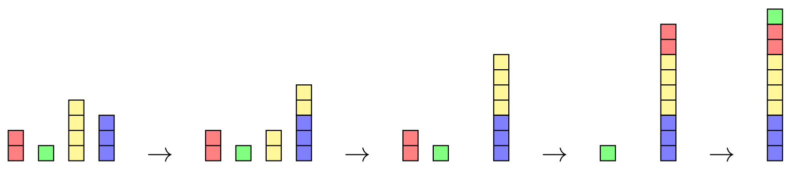
Problem B - Block stacks
UAC3 - 2015

A storage company has a robot to help them stack boxes. They have a set of k stacks of boxes and they want to make a single stack with all the boxes. The robot can carry at most b boxes at a time. You were hired to program the robot so that it can place all the boxes into a single stack as fast as possible, that is, with the minimum number of operations. An operation consists of going to a stack, taking out at most b boxes from the top of the stack and placing them onto some other stack.

Given the initial size of each of the k stacks, $s_1, s_2, \ldots, s_k$, program the robot to move all the boxes into a single stack using the minimum number of operations possible.

Example:

Suppose $s_1 = 2, s_2 = 1, s_3 = 4, s_4 = 3$ and $b = 2$ so that the robot can carry at most two boxes at a time. The optimal solution uses 4 moves. One possible way of doing it is as described in the figure.



Input

The first line of the input contains a single line with two integers k and b separated by a single space. Then follows one line with k integers containing the initial size of all the stacks $s_1, s_2, \ldots, s_k$.

Limits

- $1 \leq k \leq 10^5$
- $1 \leq s_i \leq 10^3$
- $1 \leq b \leq 10^3$

Output

A single line with the minimum number of operations the robot needs to perform in order to move all the boxes into a single stack.

Sample Test Cases

Sample Input 1

```
3 3
3 3 3
```

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
4
```

Problem C - Cash trouble
UAC3 - 2015

A group of n friends went camping. During their camp, every time they went out together they kept track of who paid money for the various purchases. By the end of the holiday they had a huge list of transactions of the form: person A owes person B, x euros.

They would like to avoid having to handle a lot of cash. For that reason, they want to find an alternative list of transactions that minimizes the total amout of money that is transferred and also settles all the debts. Can you help them?

Example 1:

Owes	Is owed	Amount
Alice	Bob	50
Bob	Alice	40

Alice owes 50 euros to Bob and Bob owes 40 euros to Alice. There are several ways they can settle the debts. One way would be for Alice to give 50 euros to Bob and then Bob to give 40 euros to Alice. This makes a total cash flow of 50 + 40 = 90 euros. Intead, Alice could simply give 10 euros to Bob. This also settles the debts but involves a cash flow of only 10 euros.

Example 2:

Owes	Is owed	Amount
Sally	Katie	50
Katie	Sally	150
Katie	Steven	20
Sally	Steven	100
Steven	Freddy	30
Freddy	Katie	30

The total money involved in these transactions is $50+150+20+100+30+30 = 380$ euros. However, debts can be settled simply by having Katie give 90 euros to Steve. This is the minimum amout of money that needs to be transferred in order to settle all the debts.

Input

The first line of the input contains two integers n and m giving the number of friends and the number of lines in the list. Then follow m lines each with three integers p_1, p_2 and x: the id of the person who owes money, the id the the person who is owed money and the amout of money, respectivelly.

The id's range from 1 to n.

Limits

- $1 \leq n \leq 10^3$
- $1 \leq m \leq 10^5$
- $1 \leq x \leq 10^3$
- $1 \leq p_1, p_2 \leq n$
- $p_1 \neq p_2$

Output

A single line with the minimum amout of money that needs to be transferred in order to settle all the debts.

Sample Test Cases

Sample Input 1

```
4 6
1 2 50
2 1 150
2 3 20
1 3 100
3 4 30
4 2 30
```

Sample Output 1

```
90
```

Sample Input 2

```
4 4
1 2 50
2 3 50
3 4 50
4 1 50
```

Sample Output 2

```
0
```

Problem D - Halting problem
UAC3 - 2015

In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever.

Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist.

In this problem you are given a description of a program and several inputs for that program. Your task is to decide whether or not the problem will halt on each of the given inputs.

```
n = readInt()
m = readInt()
g = array(n, List())
for(i = 1 to m) {
  o = readInt()
  d = readInt()
  w = readInt()
  g[o].add((d, w))
}
D = array(n, +oo)
D[0] = 0
Q = List()
Q.add((0, 0))
while(Q.size() > 0) {

  Sort Q by non-degreasing second coordinate
  (break ties by non-decreasing first coordinate)

  (x, d) = Q.removeFirst()
  if(D[x] != d) continue
  forall((y, w) in g[x])
    if(D[x] + w < D[y]) {
      D[y] = D[x] + w
      Q.add((y, D[y]))
    }
}
```

where

- `readInt()` reads the next integer from the input
- `array(n, v)` creates an array of size `n` with each position initialized do `v`

- `List()` represents the constructor of a list with three methods:
  - `add(e)` that adds an element
  - `removeFirst()` that removes and retrieves the first element of the list
  - `size()` that returns the number of elements in the list.

Input

The first line of the input contains two integers n and m separated by a single space. Then follow m lines each with three integers: o_i, d_i, w_i.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq m \leq n(n-1)/2$
- $0 \leq o_i, d_i < n$
- all pairs (o_i, d_i) are all distinct.
- $-2^{31} \leq w_i \leq 2^{31} - 1$

Output

A single line with `yes` if the algorithm will halt on the given input or `no` otherwise.

Sample Test Cases

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

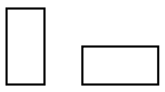
```
6 9
0 1 2
0 2 1
1 2 -4
2 3 0
3 4 1
4 3 2
4 5 1
5 1 3
5 0 -1
```

Sample Output 2

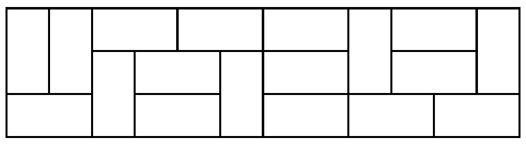
```
no
```

Problem E - Placing tiles
UAC3 - 2015

In how many ways can you fill a $3 \times n$ rectangle with 2×1 dominoes? The 2×1 dominoes are the following:



Here is a sample tiling of a 3×12 rectangle.



Input

The input contains a single integer n.

Constraints

- $1 \leq n \leq 30$

Output

A single integer giving the number of possible tilings of a $3 \times n$ rectangle.

Sample Test Cases

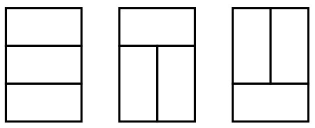
Sample Input 1

```
2
```

Sample Output 1

```
3
```

All solutions for $n = 2$:



Sample Input 2

```
8
```

Sample Output 2

```
153
```