Problem A. Data Aggregator

Time limit: 1 second

As a data scientist, you frequently process some data from experiments.

To ease your work and make your life better, you plan to write a program to compute the sum, maximum and minimum of a sequence of integers. How fast can you write this program?

Input

The first line of the input contains a single integer n $(1 \le n \le 10)$, denoting the number of integers in the sequence.

The second line contains these n integers $a_1, a_2, \cdots, a_n$ $(1 \le a_i \le 100)$.

Output

Output three lines: the sum, the maximum, and the minimum of the given integers.

Sample Input 1	Sample Output 1
3 1 3 2	6 3 1

This page is intentionally left blank.

Problem B. HEMU

Time limit: 1 second

HIT Emulator (HEMU) is an emulator for s86 architecture. The s86 architecture is stack-based, which means machine instruction can only operate on the top of a stack. The computational model for s86 contains a stack and a finite-length program. Every element of the stack is a 64-bit unsigned integer. The program may contain instructions listed in Table 1, and must end with `end` instruction. When the s86 machine starts, the stack is initialized to empty, then the machine will execute the instructions of the program in order, until the last `end` instruction is executed. After the execution of the `end` instruction, the machine halts and prints the top element of the stack.

Mnemonic	Function	Limitation
`p1`	Push constant 1 to the stack.	none
`dup`	Push a copy of the top element to the stack.	$\lvert S\rvert > 0$
`pop`	Remove the top element of the stack.	$\lvert S\rvert > 0$
`swap`	Exchange the top two elements of the stack.	$\lvert S\rvert \geq 2$
`add` x	Add the x-th element to the top element.	$0 \leq x < \lvert S\rvert$
`sub` x	Subtract the x-th element from the top element.	$0 \leq x < \lvert S\rvert$
`mul` x	Multiply the top element by the x-th element.	$0 \leq x < \lvert S\rvert$
`end`	Print the top element and halt the machine.	$\lvert S\rvert > 0$; must be the last instruction

Table 1: Instruction set for s86.

In the above table, $|S|$ refers to the size of the current stack; "the x-th element" refers to the x-th element from the top to the bottom of the stack. The top element itself is the 0-th element.

Notice that, in s86, all arithmetic instructions (`add`, `sub`, `mul`) are modulo 2^{64}. In other words, if the arithmetic result is X, the result of the s86 instruction is the unique X' such that $0 \leq X' < 2^{64}$ and $X - X'$ is a multiple of 2^{64}.

Now, the development of HEMU is being finished. To test the correctness of HEMU, please write an s86 program to print a specific integer.

Input

The input contains a single integer N ($0 \leq N < 2^{64}$).

Output

Print a s86 program of **at most 150 instructions**, one instruction per line, such that the machine prints N after execution. Note that any violation of limitation in Table 1 when executing the corresponding instruction renders your answer wrong.

Sample Input 1	Sample Output 1
2	p1 add 0 end

Sample Input 2

```
15
```

Sample Output 2

```
p1
dup
add 1
add 1
dup
add 2
add 2
mul 1
swap
pop
end
```

Sample Input 3

```
0
```

Sample Output 3

```
p1
dup
sub 1
end
```

Sample Input 4

```
18446744073709551615
```

Sample Output 4

```
p1
dup
add 1
swap
sub 1
end
```

Note

The execution of the second running example is shown below.

Instruction	Stack
p1	[1)
dup	[1 1)
add 1	[1 2)
add 1	[1 3)
dup	[1 3 3)
add 2	[1 3 4)
add 2	[1 3 5)
mul 1	[1 3 15)
swap	[1 15 3)
pop	[1 15)
end	(print 15)

Table 2: Running example of an s86 program.

Problem C. Majority 3-SAT

Time limit: 1 second

Assume there are k Boolean variables $x_1, x_2, \cdots, x_k$. We say a triplet (a, b, c) (where a, b, or c can be a Boolean variable or the negation of a Boolean variable) is satisfied, if at least two of a, b, c are true.

Given n triplets, you have to decide if there is an assignment to these k Boolean variables, such that all these n triplets are satisfied.

Input

The first line of the input is a single integer T $(1 \le T \le 10)$, denoting the number of test cases.

Each test case starts with a line of two integers k, n $(1 \le k, n \le 10^4)$, denoting the number of Boolean variables and the number of triplets. Then follow n lines, describing the n triplets. Each of these n lines contains three integers a, b, c $(1 \le |a|, |b|, |c| \le k)$, denoting a triplet. Take a as an example: if $a > 0$, it denotes the variable x_a; if $a < 0$, it represents the negation of x_{-a}. It is the same for b and c.

Output

For each test case, if there exists an assignment such that alll the triplets are satisfied, print `yes` in one line; otherwise, print `no` in one line.

Sample Input 1

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

Sample Output 1

```
yes
no
yes
no
```

Note

For the first sample data, you may let x_1, x_3 be true, x_2 be false, then both triplets are satisfied.

This page is intentionally left blank.

Problem D. Interval Covering

Time limit: 2 seconds

There are n intervals in a number axis. The i-th interval is initially $[l_i, r_i]$, and you may pay c_i to extend the interval in either direction for one unit. For example, given an interval $[4, 6]$, after extending for one unit, the interval may become $[3, 6]$ or $[4, 7]$.

Given the initial intervals and the unit cost for extending every interval, please compute the minimum cost to extend these intervals such that they cover $[0, m]$.

Input

The first line of the input contains two integers n, m ($1 \le n \le 2000, 0 \le m \le 10^9$), as specified in problem statement.

The next n lines describe the n intervals. The i-th of them contains three integers l_i, r_i, c_i ($0 \le l_i \le r_i \le m, 1 \le c_i \le 10^6$), denoting the endpoints and the unit extension cost of the i-th interval.

Output

Output a single integer, denoting the minimum total cost.

Sample Input 1	Sample Output 1
2 5 1 2 3 4 5 1	4

Sample Input 2	Sample Output 2
3 7 0 0 6 3 5 1 6 7 2	4

This page is intentionally left blank.