Task GCD 2

⏳ 4.5 sec. 💾 512 MB

Who does not love machine code? 🙂

You are given an array $a_1, a_2, \ldots, a_N$, initially filled with the values 0. You are also given Q queries of two types:

1. Add x to $a_l, a_{l+1}, \ldots, a_r$.
2. Print $\sum_{i=l}^{r} a_i$.

Write a program which answers the queries.

Implementation details

The program that you will write must be written in pseudo machine code in a text file (`txt`). Your code can use the registers listed below, as well as a helper array $m_0, m_1, \ldots, m_{10^6-1}$. The array and the registers hold 64-bit integers.

Let us denote by $|R|$ the value of register R. The language supports the following instructions:

```
IN R
```

This instruction reads a number into register R.

```
OUT R
```

This instruction prints the number in register R.

```
MOV R1 R2
```

This instruction copies $|R2|$ into register $R1$.

```
MOV R1 x
```

This instruction, used in this way, stores the number x in register $R1$.

```
ADD R1 R2
```

This instruction adds $|R2|$ to $|R1|$ and the result stays in $R1$.

```
SUB R1 R2
```

This instruction subtracts $|R2|$ from $|R1|$ and the result stays in $R1$.

```
DIV R1 R2
```

This instruction divides $|R1|$ by $|R2|$ using integer division and the result stays in $R1$.

```
MUL R1 R2
```

This instruction multiplies $|R1|$ by $|R2|$ and the result stays in $R1$.

```
CMP R1 R2
```

This instruction compares $|R1|$ and $|R2|$ and stores the result in register RES. The result is 0 if $|R1| > |R2|$ and 1 if $|R1| \leq |R2|$.

```
LOAD R1 R2
```

This instruction loads $m_{|R1|}$ into register $R2$.

```
STORE R1 R2
```

This instruction stores $|R2|$ in $m_{|R1|}$.

```
JZ N
```

This instruction tells the interpreter to go to line N in your code, only if $|RES| = 0$.

```
JNZ N
```

This instruction tells the interpreter to go to line N in your code, only if $|RES| \neq 0$.

```
RETURN
```

This instruction terminates the execution of the program.

```
; comment
```

```
MOV A B ; comment
```

You can use comments in your code by using a semicolon.

- $R, R1, R2$: a register. It can be A, B, C, D, E, F, G, H or RES. You are allowed to write to all of them.
- N: a line number in the code. The indexing starts from 1.

Input

The first line of the input contains N and Q. Each of the next Q lines contains t, l, r, where t is the type of the query. If $t = 1$, the number x is also given at the end of the line.

Output

For each query of type 2 print the answer.

Constraints

- $1 \leq N, Q \leq 3.10^4$
- $1 \leq l \leq r \leq N$
- $1 \leq x \leq 10^9$
- $1 \leq t \leq 2$

Subtasks

Subtask	Points	Required subtasks	Additional constraints
1	10	–	$N = 1, Q \leq 500$
2	35	1	$N, Q \leq 500$
3	60	1	For queries of type 1, $l = r$.
4	45	$1 - 3$	–

Sample program

Let us look at a sample program in this language, which reads two numbers and prints their sum:

```
IN A
IN B
ADD A B
OUT A
```

Let us look at another sample program, which reads two numbers and prints the greater of them:

```
IN A
IN B
CMP A B
JZ 6
JNZ 8
OUT A
RETURN
OUT B
RETURN
```

Sample grader

The provided grader reads the code from a file `nod2.txt` and executes it.