Register Machine (machine)

During renovations of the Bock Casemates – a network of underground fortifications in Luxembourg City – workers have dug up an ancient treasure chest, sealed by a rather peculiar mechanism.

The inscription on the lid calls it a *register machine*, and explains that it has 64 registers numbered 0 to 63, each of which can hold a nonnegative integer. Initially, all registers contain 0.

A collection of knobs and buttons allows you to input a *program* consisting of a sequence of instructions. The first instruction you input is labelled 0, the next instruction 1, and so on. There are three kinds of instruction:

- `INCREMENT x, i` — The value stored in register x is increased by 1, then the machine processes the instruction labelled i.
- `DECREMENT x, i, j` — If the value stored in register x is 0, the machine processes the instruction labelled i. Otherwise, register x is decreased by 1, then the machine processes the instruction labelled j.
- `HALT` — The machine stops processing instructions.

The machine starts processing from the instruction labelled 0. Once it stops, it will open the chest if the value stored in register 0 is exactly N.

If it does not stop after processing 50 000 000 instructions, or if it attempts to process an instruction that does not exist, the machine will instead catastrophically break down and seal the chest forever.

Your task is to enter a program that will open the chest. Since the machine is delicate, you want the length of the program to be as short as possible, and certainly no more than 100 instructions.

Implementation

You will have to submit a single `.cpp` source file.

⚠ Among this task's attachments you will find a template `machine.cpp` and a sample grader `grader.cpp` you can use to test your solutions locally.

You will have to implement the following function:

C++	`void program_machine(int N)`

- Integer N represents the number to store in register 0 when the machine halts.

Your program can call the following functions at most 100 times, which are already defined in the grader:

C++	`void add_increment(int x, int i)`
C++	`void add_decrement(int x, int i, int j)`
C++	`void add_halt()`

Sample Grader

Among this task's attachments you will find a simplified version of the grader used during evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin`, calls the function `program_machine` and writes back on `stdout` using the following format.

The input file consists of one line, containing the integer N.

The output is made up of one line:

- If your program opens the chest correctly, the output is `OK L`, where L is the length of your program.
- Otherwise, the output begins with `WA`, followed by a short reason (e.g. program exceeded 100 instructions, program did not stop in time, etc.)

Constraints

- $1 \leq N \leq 1\,000\,000$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program must correctly open the chest on every test case in that subtask.

For subtask 3, the score is determined by the length of your program. Let L be the maximum length of your program across all test cases in this subtask. The score for this subtask is $\min(77, 100 - L)$. In particular, to obtain the full 77 points for this subtask, your program must use at most 23 instructions on all test cases for this subtask.

- **Subtask 0 [0 points]**: Sample test cases.
- **Subtask 1 [7 points]**: $N \leq 99$.
- **Subtask 2 [16 points]**: N is a power of 2 (i.e. $N = 2^k$ for some integer $k \geq 0$).
- **Subtask 3 [77 points]**: No additional constraint. (Note the scoring for this subtask depends on the length of your program.)

Examples

Grader

```
program_machine(3)
```

Solution

```
add_increment(1, 1)
add_increment(0, 2)
add_decrement(1, 4, 1)
add_halt()
add_increment(0, 3)
```

Grader

```
program_machine(5)
```

Solution

```
add_increment(0, 1)
add_increment(0, 2)
add_increment(0, 3)
add_increment(0, 4)
add_increment(0, 5)
add_halt()
```

Explanation

In the first sample, $N = 3$. The example creates a program with 5 instructions:

0.	INCREMENT 1, 1
1.	INCREMENT 0, 2
2.	DECREMENT 1, 4, 1
3.	HALT
4.	INCREMENT 0, 3

The following table shows the label of each instruction the machine processes and the contents of registers 0 and 1 afterwards:

Instruction	**Register 0**	**Register 1**
—	0	0
0	0	1
1	1	1
2	1	0
1	2	0
2	2	0
4	3	0
3	3	0

In the second example, $N = 5$ and the example program shown uses 6 instructions.