

Problem B – The Halting Problem

Input file: **standard input**
Output file: **standard output**
Time limit: **3 seconds**
Memory limit: **256 megabytes**

Today, your task is to solve the halting problem.

You have heard that this is impossible in general, but perhaps it would be possible in a simpler programming language.

After spending a term writing a compiler for WLP4, you decide to create its successor, WLP5. You get a little carried away designing new language features, and by the time you announce the language to the world, it has quite a few different instructions. A few people even started writing programs in WLP5!

Unfortunately, you still have to write the compiler.

With your time running short, you start cutting features. Eventually, only two instructions, **GOTO** and **HALT**, remain. At least the compiler should be easy to finish now. You might even have time to tackle the halting problem!

But the flourishing WLP5 community has already used the M instructions you removed. You decide to keep accepting them as legacy instructions, although they now do nothing and simply continue to the next instruction. Hopefully, nobody was relying on them too much.

Before releasing the compiler, you would like to know: how many WLP5 programs consisting of exactly N instructions eventually halt?

WLP5 Language Specification

A WLP5 program consists of N instructions numbered from 1 to N . Each instruction has one of the following forms:

- **GOTO** x , where $1 \leq x \leq N$: jump to instruction x .
- **LEGACY** y , where $1 \leq y \leq M$: do nothing, then continue to the next instruction.
- **HALT**: terminate execution successfully.

Execution begins at instruction 1. Formally, let i be the position of the instruction to execute next. Initially, $i = 1$. Then, repeat the following steps:

1. If $i = N + 1$, execution ends with a runtime error.
2. Otherwise, execute instruction i :
 - If it is **GOTO** x , set $i = x$.
 - If it is **LEGACY** y , set $i = i + 1$.
 - If it is **HALT**, execution terminates successfully.

A program **halts** if execution reaches a **HALT** instruction after finitely many steps. Programs that run forever or end with a runtime error are not counted.

Two programs are different if their instructions differ at any position. Instructions with different operands are different: for example, **LEGACY** 1 and **LEGACY** 2 are distinct instructions, even though they behave identically. WLP5 programs are whitespace and formatting-insensitive – those features were cut too.

Count the number of halting WLP5 programs consisting of exactly N instructions. Since this number may be very big, output it modulo the large prime 998 244 353.

Input

The only line of input contains two space-separated integers N and M ($1 \leq N \leq 5000$, $1 \leq M \leq 10^8$).

Output

Output the number of halting WLP5 programs consisting of exactly N instructions, modulo 998 244 353.

Examples

standard input	standard output
1 3	1
2 2	8
3 1	45