Topic 2: Combinatorics: Review, Linear Recurrence **Lecture Notes**
CS 41100 - CP3 Competitive Programming III (Spring 2024) **Purdue University**
Instructor: Zhongtang Luo **Date:** February 3, 2024
Notes Preparer: Zhongtang Luo

1 Look for Patterns!

1.1 Become OEIS

> **Sample Problem:** Pyramid
> **Link:** `https://vjudge.net/problem/CodeForces-101981G`

In problems like these, we need to observe that there is one (or two) input(s) and one output. Maybe we can bash possible sequences in our head! (or in a non-ICPC style contest, refer to OEIS)

Common sequences to look for are listed below:

- Binomial Coefficients (look at the various identities listed)
- Pascal's Triangle
- Catalan Numbers
- Stirling Numbers
- A common trick while recognizing sequences is to look at differences and possibly, difference of differences.

Something that could help is simply listing out the first 10-15 numbers in the above sequences (write code to generate more!) and checking for possible patterns that correlate to the sequence.

There are a lot of ways to recognize the sequence listed in this particular problem but the simplest sequence that could be found is just $\binom{n+3}{4}$. (Look out for modular arithmetic in this problem!)

An AC solution to the problem: Submission

1.2 Or Cheat (kinda!)

There is a complex algorithm that can find out a linear recurrence (if it exists) associated with the given sequence.

The description of the algorithm can be found here: Berlekamp-Massey Algorithm.
The template for the algorithm can be found here: Template.

Try plugging the annoying sequence into this algorithm in dire situations before giving up!

2 Basic Combinatorics

2.1 What You Need to Know

A brief list of topics that show up (semi-)regularly is listed here:

- Power, Fast power, Inverse
- Permutation, Combination, Binomial coefficient
- Stars and Bars

The sequences listed in Section 1.1 also contain useful identites that come up often!

Helpful resources on the algorithms/concepts described above are these CF blogs:

- Part 1 - addition / multiplication principle, factorial, permutation and combination, special binomial theorem, principle of inclusion-exclusion, pigeonhole principle
- Part 2 - Quick power, Fermat's little theorem, extend-gcd, multiplicative inverse of an integer

2.2 Observational Problem involving Combinatorics

Sample Problem: Doremy's Pegging Game
Link: `https://vjudge.net/problem/CodeForces-1764D`

This problem involves making a string of observations which will lead to a combinatorial closed form for the answer. The solution used here is adapted from this comment on the editorial for the problem.

(There is also an O(n) solution to the problem listed here.)

The first observation we should make is that we can use symmetry (which is common in problems involving games/operations on circles).
Let us observe the ending state. The rubber band would be stretched over the blue peg, some corner peg a and the other corner peg b, forming a sector of the circle with possibly some pegs not used.
So, we can arbitrarily start finding out a closed form by fixing peg a to be peg 0. Then, before outputting the answer, we can multiply it by n as we would have computed the same closed form if we had fixed any other peg to be peg a than peg 0.

Now, we obviously have to remove all pegs between peg b and peg $n-1$.

We can choose to remove any pegs between $[1, b-1]$ as they don't change our overall structure.

Another observation to be made is that peg b (given that peg a is fixed to be peg 0) must be in the range $[0, \lceil \frac{n}{2} \rceil - 1]$. This can be explained by thinking of peg $\lceil \frac{n}{2} \rceil$ as diametrically opposite to peg 0 which would mean plucking any peg after $\lceil \frac{n}{2} \rceil$ would collapse the rubber band onto the blue peg.

The last peg to be removed using this "diametrically opposite" way of thinking must lie in the range $[\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} + b \rfloor]$.

Now, we have found all the observations that we need to lead us to the final result. The final algorithm is now as follows:

- Efficiently precompute factorials and binomial coefficients.
- Initialize the answer to 0.
- Start at peg 0 (as peg a) and enumerate all possible peg b's.
- With this structure of pegs a and b, find out the number of pegs that must to be removed, pegs that are optional to be removed and the last peg that is removed.
- Enumerate the number of optional pegs we remove.
- Now, if we choose o optional pegs from the total O optional pegs, the number of ways to pick an order of all pegs to be removed before the last peg is $\binom{O}{o} \times (m+o)!$ where m is the number of pegs that must be removed.
- We also have l pegs that can be the last peg to be removed. However, these l pegs are included in the $m+o$ pegs. So, we multiply the above $\binom{O}{o} \times (m+o)!$ by $\frac{l}{m+o}$. Add this to the answer.
- Once we're done with this, output the answer multiplied by n.

An AC solution to this problem: Submission

3 Ability to List Formulae

Sample Problem: Sky Full of Stars
Link: `https://vjudge.net/problem/CodeForces-997C`

For this problem, we just need to have the ability to decompose the problem into disjoint combinatorial formulae.

Here, we want to find out the number of grid colorings with at least one row and one column both. This can be decomposed into grids which have at least one row colored, grids which have at least one column colored and grids which have both. Now, this seems like the perfect problem to count and use inclusion-exclusion.

Call grids with at least one column colored S_1, grids with at least one row colored S_2 and grids with at least one row and at least one column colored S_3.

By symmetry, $S_1 = S_2$. The final answer is, thus, $S_1 + S_2 - S_3$ by inclusion-exlusion principle.

$$S_1 = S_2 = \sum_{i=1}^{n} 3^i \times 3^{n\times(n-1)} \times \binom{n}{i} \times (-1)^{i-1}$$

$$S_3 = 3 \times \sum_{i=1}^{n}\sum_{j=1}^{n} 3^{(n-i)\times(n-j)} \times \binom{n}{i} \times \binom{n}{j} \times (-1)^{i+j}$$

However, calculating S_3 takes $\mathcal{O}(n^2)$ time to compute which is too slow. However, we can make an observation to speed this up.

$$S_3 = 3 \times \sum_{i=1}^{n} 3^{(n-i)} \times \binom{n}{i} \times (-1)^i \sum_{j=1}^{n} 3^{(n-j)} \times \binom{n}{j} \times (-1)^j$$

$$S_3 = 3 \times \sum_{i=1}^{n} 3^{(n-i)} \times \binom{n}{i} \times (-1)^i \times ((3-1)^n - 3^n)$$

Now, we can compute all of these in $\mathcal{O}(n)$ time.

An AC submission implementing a different (maybe easier idea) is this: Submission. This implements the idea from this comment.

4 Linear Recurrence

A lot of problems can be decomposed into a linear recurrence similar to the form:

$$f(i) = \sum_{j=1}^{k} c_j \times f(i-j)$$

There is a way to solve these recurrences quick and easily: Matrix Exponentiation. A helpful blog on this topic is: Guide on Matrix Exponentiation.

This technique can help us solve easier problems like computing Fibonacci numbers quickly as well as solve harder problems like finding the number of paths with length k in an unweighted undirected graph.

4.1 Spoilers: this is not so easy!

> **Sample Problem:** So Easy!
> **Link:** `https://vjudge.net/problem/HDU-4565`

For this problem, we see that we can easily compute $S_1 = a + \sqrt{b}$. Now, we can express S_n in a linear recurrence and use matrix exponentiation to find out S_n. Observe:

$$S_{n+1} = -2a \times S_n + (a^2 - b) \times S_{n-1}$$

We can prove that this works as follows:

$$\begin{aligned} S_{n+1} &= (a + \sqrt{b})S_n \\ 2a \times S_n &= (a + \sqrt{b})S_n + (a - \sqrt{b})S_n \\ &= S_{n+1} + \frac{a^2 - b}{a + \sqrt{b}} S_n \\ &= S_{n+1} + (a^2 - b)S_{n-1} \\ \implies S_{n+1} &= -2a \times S_n + (a^2 - b) \times S_{n-1} \end{aligned}$$

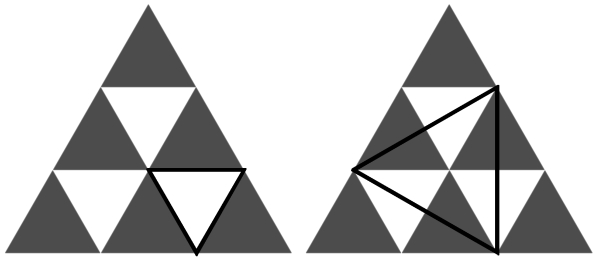
And this is a linear recurrence that can be solved by matrix exponentiation.

Pyramid

The use of the triangle in the New Age practices seems to be very important as it represents the unholy trinity (Satan the Antichrist and the False Prophet bringing mankind to the New World Order with false/distorted beliefs). The triangle is of primary importance in all Illuminati realms whether in the ritual ceremonies of the Rosicrucians and Masons or the witchcraft astrological and black magic practices of other Illuminati followers.

One day you found a class of mysterious patterns. The patterns can be classified into different degrees. A pattern of degree n consists of $n(n+1)/2$ small regular triangles with side length of 1 all in the same direction. The figure below shows the pattern of degree 3. All small regular triangles are highlighted.

Since the pattern contains many regular triangles which is very evil and unacceptable, you want to calculate the number of regular triangles formed by vertices in the pattern so that you can estimate the strength of Illuminati. It is not necessary that each side of regular triangles is parallel to one side of the triangles. The figure below shows two regular triangles formed by vertices in a pattern of degree 3.



Since the answer can be very large, you only need to calculate the number modulo $10^9 + 7$.

Input

The first line contains an integer t ($1 \le t \le 10^6$) — the number of test cases. Each of the next t lines contains an integer n ($1 \le n \le 10^9$) — the degree of the pattern.

Output

For each test case, print an integer in one line — the number of regular triangles modulo $10^9 + 7$.

Examples

Input

```
6
1
2
3
```

```
4
5
6
```

Output

```
1
5
15
35
70
126
```

Source

2018-2019 ACM-ICPC Asia Nanjing Regional

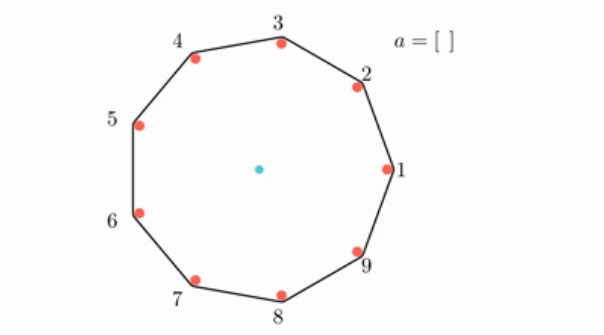
Doremy's Pegging Game

Doremy has $n + 1$ pegs. There are n red pegs arranged as vertices of a regular n-sided polygon, numbered from 1 to n in anti-clockwise order. There is also a blue peg of **slightly smaller diameter** in the middle of the polygon. A rubber band is stretched around the red pegs.

Doremy is very bored today and has decided to play a game. Initially, she has an empty array a. While the rubber band does not touch the blue peg, she will:

1. choose i ($1 \le i \le n$) such that the red peg i has not been removed;
2. remove the red peg i;
3. append i to the back of a.

Doremy wonders how many possible different arrays a can be produced by the following process. Since the answer can be big, you are only required to output it modulo p. p is guaranteed to be a prime number.



game with $n = 9$ and $a = [7, 5, 2, 8, 3, 9, 4]$ and another game with $n = 8$ and $a = [3, 4, 7, 1, 8, 5, 2]$

Input

The first line contains two integers n and p ($3 \le n \le 5000$, $10^8 \le p \le 10^9$) — the number of red pegs and the modulo respectively.

p is guaranteed to be a prime number.

Output

Output a single integer, the number of different arrays a that can be produced by the process described above modulo p.

Examples

Input

```
4 100000007
```

Output

```
16
```

Input

```
1145 141919831
```

Output

```
105242108
```

Note

In the first test case, $n = 4$, some possible arrays a that can be produced are $[4, 2, 3]$ and $[1, 4]$. However, it is not possible for a to be $[1]$ or $[1, 4, 3]$.

Source

Codeforces Global Round 24

Sky Full of Stars

On one of the planets of Solar system, in Atmosphere University, many students are fans of bingo game.

It is well known that one month on this planet consists of n^2 days, so calendars, represented as square matrix n by n are extremely popular.

Weather conditions are even more unusual. Due to the unique composition of the atmosphere, when interacting with sunlight, every day sky takes one of three colors: blue, green or red.

To play the bingo, you need to observe the sky for one month — after each day, its cell is painted with the color of the sky in that day, that is, blue, green or red.

At the end of the month, students examine the calendar. If at least one row or column contains only cells of one color, that month is called lucky.

Let's call two colorings of calendar different, if at least one cell has different colors in them. It is easy to see that there are $3^{n \cdot n}$ different colorings. How much of them are lucky? Since this number can be quite large, print it modulo 998244353.

Input

The first and only line of input contains a single integer n ($1 \le n \le 1000\,000$) — the number of rows and columns in the calendar.

Output

Print one number — number of lucky colorings of the calendar modulo 998244353

Examples

Input

```
1
```

Output

```
3
```

Input

```
2
```

Output

```
63
```

Input

```
3
```

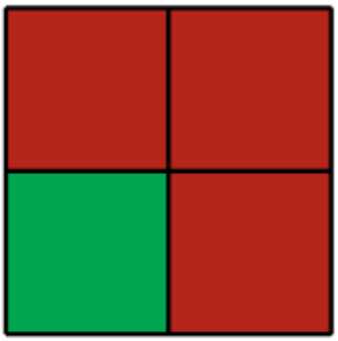
Output

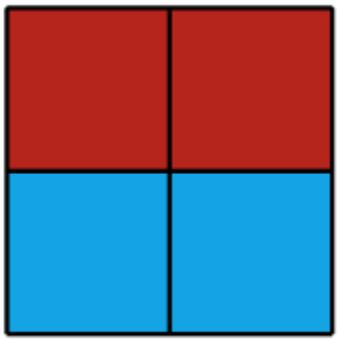
```
9933
```

Note

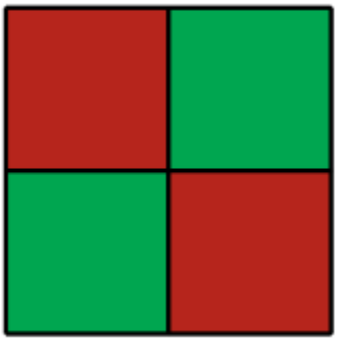
In the first sample any coloring is lucky, since the only column contains cells of only one color.

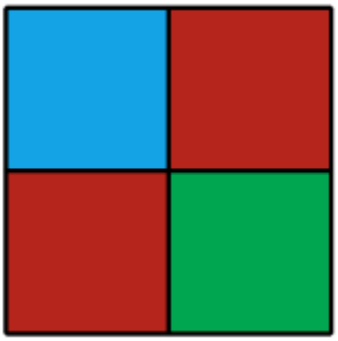
In the second sample, there are a lot of lucky colorings, in particular, the following colorings are lucky:





While these colorings are not lucky:





Source

Codeforces Round 493 (Div. 1)

So Easy!

A sequence S_n is defined as:

$$S_n = \left\lceil (a + \sqrt{b})^n \right\rceil \%m$$

where a, b, n, and m are positive integers. $\lceil x \rceil$ is the ceiling of x. For example, $\lceil 3.14 \rceil = 4$. You, a top coder, are tasked with calculating S_n.

You, a top coder, say: "So easy!"

Input

There are several test cases, each test case in one line contains four positive integers: a, b, n, m. Where $0 < a, m < 2^{15}$, $(a-1)^2 < b < a^2$, $0 < b, n < 2^{31}$. The input will finish with the end of the file.

Output

For each case, output an integer S_n.

Examples

Input

```
2 3 1 2013
2 3 2 2013
2 2 1 2013
```

Output

```
4
14
4
```

Source

2013 ACM-ICPC Changsha Invitational