Xp Orbs

In Minecraft, for every task completed, the player is rewarded with a certain number of experience points in the form of some green orbs, each orb rewarding the player with different amounts of experience based on its size.

An orb of size i rewards the player with xp_i experience points. Where xp is defined as follows:

- $xp_1 = 1$;
- $xp_i = prev\_prime(2 \cdot xp_{i-1})$, where $prev\_prime(a)$ is the largest prime number that is smaller than or equal to a. For example, $prev\_prime(16) = 13$ and $prev\_prime(23) = 23$.

For instance, the first 8 sizes of orbs reward the player with: $1, 2, 3, 5, 7, 13, 23$ and 43 experience points, respectively.

Notch, the creator of Minecraft, made it so that any non-negative integer number of experience points can be broken down as a sum of experience rewarded by orbs in the following way (here $\oplus$ represents array concatenation):

- Let $dec(a)$ be an array representing the decomposition of a experience points as a sum of experience rewarded by orbs;
- $dec(0) = [\,]$ (the empty array)
- $dec(a) = [xp_{max}] \oplus dec(a - xp_{max})$, where xp_{max} is the largest element in xp such that $xp_{max} \leq a$. For example, the decomposition of 11 is $dec(11) = [7, 3, 1]$ and the decomposition of 15 is $dec(15) = [13, 2]$. He also defined $cnt(a)$ to be the length of the array $dec(a)$, therefore $cnt(11) = 3, cnt(15) = 2$.

Notch wants to know the answer to q queries of the following form:

- l, r – find the sum $\dfrac{l}{cnt(l)} + \dfrac{l+1}{cnt(l+1)} + \ldots + \dfrac{r-1}{cnt(r-1)} + \dfrac{r}{cnt(r)}$

Input

The first line contains a single integer representing the number of queries q. Each of the next q lines contains a pair of integers. The i^{th} of these lines describes the i^{th} query: l_i and r_i.

Output

The output contains q lines. The i^{th} of these lines contains a single integer representing the answer to the i^{th} query.

Note regarding printing the output. Let the fraction $\frac{x}{y}$ be the answer for a query. In order to output it, you should print a single integer representing the product $x \cdot mod\_inv(y) \ mod \ 998\ 244\ 353$, where $mod\_inv(y)$ is defined as $mod\_inv(y) = y^{998\ 244\ 351} \ mod \ 998\ 244\ 353$.

Note regarding modular arithmetic. Aditionally, keep in mind the following:

- Given two fractions $\frac{a}{b}$ and $\frac{c}{d}$, their modular sum can be easily computed as: $(a \cdot mod\_inv(b) + c \cdot mod\_inv(d)) \ mod \ 998\ 244\ 353$;
- If two fractions $\frac{a}{b}$ and $\frac{c}{d}$ are equal, then $a \cdot mod\_inv(b) \ mod \ 998\ 244\ 353 = c \cdot mod\_inv(d) \ mod \ 998\ 244\ 353$.

Constraints

- $1 \leq q \leq 5 \cdot 10^4$
- $1 \leq l_i \leq r_i \leq 10^{12}$

Subtasks

#	Points	Restrictions
1	18	$0 \leq r_i - l_i < 100$
2	65	$1 \leq l_i \leq r_i \leq 10^8$
3	17	No further constraints.

Examples

Input Example #1

```
2
5 12
1 1000000
```

Output Example #1

```
166374097
439931963
```

Input Example #2

```
5
11 15
5 14
3 10
12 20
7 19
```

Output Example #2

```
166374096
166374117
499122210
499122249
665496322
```

Explanation

For the first query in the first example, the answer, starting with $ans = 0$, can be computed as follows:

- $dec(5) = [5] \rightarrow ans \mathrel{+}= \frac{5}{1}$
- $dec(6) = [5, 1] \rightarrow ans \mathrel{+}= \frac{6}{2}$
- $dec(7) = [7] \rightarrow ans \mathrel{+}= \frac{7}{1}$
- $dec(8) = [7, 1] \rightarrow ans \mathrel{+}= \frac{8}{2}$
- $dec(9) = [7, 2] \rightarrow ans \mathrel{+}= \frac{9}{2}$
- $dec(10) = [7, 3] \rightarrow ans \mathrel{+}= \frac{10}{2}$
- $dec(11) = [7, 3, 1] \rightarrow ans \mathrel{+}= \frac{11}{3}$
- $dec(12) = [7, 5] \rightarrow ans \mathrel{+}= \frac{12}{2}$

The total sum is $ans = \frac{229}{6}$ and the output is: $229 \cdot mod\_inv(6)\ mod\ 998\ 244\ 353 = 229 \cdot 166\ 374\ 059\ mod\ 998\ 244\ 353 = 166\ 374\ 097$.