A. Find Divisible

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a range of positive integers from l to r.

Find such a pair of integers (x, y) that $l \le x, y \le r$, $x \ne y$ and x divides y.

If there are multiple answers, print any of them.

You are also asked to answer T independent queries.

Input

The first line contains a single integer T ($1 \le T \le 1000$) — the number of queries.

Each of the next T lines contains two integers l and r ($1 \le l \le r \le 998244353$) — inclusive borders of the range.

It is guaranteed that testset only includes queries, which have at least one suitable pair.

Output

Print T lines, each line should contain the answer — two integers x and y such that $l \le x, y \le r$, $x \ne y$ and x divides y. The answer in the i-th line should correspond to the i-th query from the input.

If there are multiple answers, print any of them.

Example

input

```
3
1 10
3 14
1 10
```

output

```
1 7
3 9
5 10
```

B. Substring Removal

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string s of length n consisting only of lowercase Latin letters.

A substring of a string is a contiguous subsequence of that string. So, string "`forces`" is substring of string "`codeforces`", but string "`coder`" is not.

Your task is to calculate the number of ways to remove **exactly** one substring from this string in such a way that **all** remaining characters are **equal** (the number of distinct characters either zero or one).

It is guaranteed that there is **at least two** different characters in s.

Note that you **can** remove the whole string and it is correct. Also note that you should **remove at least one character**.

Since the answer can be rather large (not very large though) print it modulo 998244353.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the length of the string s.

The second line of the input contains the string s of length n consisting only of lowercase Latin letters.

It is guaranteed that there is **at least two** different characters in s.

Output

Print one integer — the number of ways modulo 998244353 to remove **exactly** one substring from s in such way that **all** remaining characters are **equal**.

Examples

input

```
4
abaa
```

output

```
6
```

input

```
7
aacdeee
```

output

```
6
```

input

```
2
az
```

output

```
3
```

Note

Let $s[l; r]$ be the substring of s from the position l to the position r inclusive.

Then in the first example you can remove the following substrings:

- $s[1; 2]$;
- $s[1; 3]$;
- $s[1; 4]$;
- $s[2; 2]$;
- $s[2; 3]$;
- $s[2; 4]$.

In the second example you can remove the following substrings:

- $s[1; 4]$;
- $s[1; 5]$;
- $s[1; 6]$;
- $s[1; 7]$;
- $s[2; 7]$;
- $s[3; 7]$.

In the third example you can remove the following substrings:

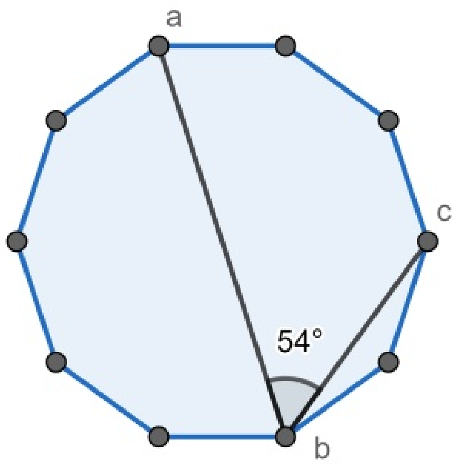
- $s[1; 1]$;
- $s[1; 2]$;
- $s[2; 2]$.

C. Polygon for the Angle

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an angle ang.

The Jury asks You to find such **regular** n-gon (regular polygon with n vertices) that it has three vertices a, b and c (they can be non-consecutive) with $\angle abc = \mathrm{ang}$ or report that there is no such n-gon.



If there are several answers, print the **minimal** one. It is guarantied that if answer exists then it doesn't exceed 998244353.

Input

The first line contains single integer T ($1 \le T \le 180$) — the number of queries.

Each of the next T lines contains one integer ang ($1 \le \text{ang} < 180$) — the angle measured **in degrees**.

Output

For each query print single integer n ($3 \le n \le 998244353$) — minimal possible number of vertices in the regular n-gon or -1 if there is no such n.

Example

input

```
4
54
50
2
178
```

output

```
10
18
90
180
```

Note

The answer for the first query is on the picture above.

The answer for the second query is reached on a regular 18-gon. For example, $\angle v_2 v_1 v_6 = 50^{\circ}$.

The example angle for the third query is $\angle v_{11} v_{10} v_{12} = 2^{\circ}$.

In the fourth query, minimal possible n is 180 (not 90).

D. Easy Problem

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya is preparing a contest, and now he has written a statement for an easy problem. The statement is a string of length n consisting of lowercase Latin latters. Vasya thinks that the statement can be considered hard if it contains a subsequence `hard`; otherwise the statement is easy. For example, `hard`, `hzazrzd`, `haaaaard` can be considered hard statements, while `har`, `hart` and `drah` are easy statements.

Vasya doesn't want the statement to be hard. He may remove some characters from the statement in order to make it easy. But, of course, some parts of the statement can be crucial to understanding. Initially the *ambiguity* of the statement is 0, and removing i-th character increases the *ambiguity* by a_i (the index of each character is considered as it was in the original statement, so, for example, if you delete character `r` from `hard`, and then character `d`, the index of `d` is still 4 even though you delete it from the string `had`).

Vasya wants to calculate the minimum *ambiguity* of the statement, if he removes some characters (possibly zero) so that the statement is easy. Help him to do it!

Recall that subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Input

The first line contains one integer n ($1 \le n \le 10^5$) — the length of the statement.

The second line contains one string s of length n, consisting of lowercase Latin letters — the statement written by Vasya.

The third line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 998244353$).

Output

Print minimum possible *ambiguity* of the statement after Vasya deletes some (possibly zero) characters so the resulting statement is easy.

Examples

input

```
6
hhardh
3 2 9 11 7 1
```

output

```
5
```

input

```
8
hhzarwde
3 2 6 9 4 8 7 1
```

output

```
4
```

input

```
6
hhaarr
1 2 3 4 5 6
```

output

```
0
```

Note

In the first example, first two characters are removed so the result is `ardh`.

In the second example, 5-th character is removed so the result is `hhzawde`.

In the third example there's no need to remove anything.

E. The Top Scorer

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Hasan loves playing games and has recently discovered a game called TopScore. In this soccer-like game there are p players doing penalty shoot-outs. Winner is the one who scores the most. **In case of ties, one of the top-scorers will be declared as the winner randomly with equal probability.**

They have just finished the game and now are waiting for the result. But there's a tiny problem! The judges have lost the paper of scores! Fortunately they have calculated sum of the scores before they get lost and also for some of the players they have remembered a lower bound on how much they scored. However, the information about the bounds is private, so Hasan only got to know his bound.

According to the available data, he knows that his score is at least r and sum of the scores is s.

Thus the final state of the game can be represented in form of sequence of p integers $a_1, a_2, \ldots, a_p$ ($0 \le a_i$) — player's scores. Hasan is player number 1, so $a_1 \ge r$. Also $a_1 + a_2 + \cdots + a_p = s$. Two states are considered different if there exists some position i such that the value of a_i differs in these states.

Once again, Hasan doesn't know the exact scores (he doesn't know his exact score as well). So he considers each of the final states to be equally probable to achieve.

Help Hasan find the probability of him winning.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \ne 0$, $P \le Q$. Report the value of $P \cdot Q^{-1} \pmod{998244353}$.

Input

The only line contains three integers p, s and r ($1 \le p \le 100$, $0 \le r \le s \le 5000$) — the number of players, the sum of scores of all players and Hasan's score, respectively.

Output

Print a single integer — the probability of Hasan winning.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \ne 0$, $P \le Q$. Report the value of $P \cdot Q^{-1} \pmod{998244353}$.

Examples

input
2 6 3
output
124780545

input
5 20 11
output
1

input
10 30 10
output
85932500

Note

In the first example Hasan can score 3, 4, 5 or 6 goals. If he scores 4 goals or more than he scores strictly more than his only opponent. If he scores 3 then his opponent also scores 3 and Hasan has a probability of $\frac{1}{2}$ to win the game. Thus, overall he has the probability of $\frac{7}{8}$ to win.

In the second example even Hasan's lower bound on goal implies him scoring more than any of his opponents. Thus, the resulting probability is 1.

F. Inversion Expectation

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A permutation of size n is an array of size n such that each integer from 1 to n occurs exactly once in this array. An inversion in a permutation p is a pair of indices (i, j) such that $i > j$ and $a_i < a_j$. For example, a permutation $[4, 1, 3, 2]$ contains 4 inversions: $(2, 1)$, $(3, 1)$, $(4, 1)$, $(4, 3)$.

You are given a permutation p of size n. However, the numbers on some positions are replaced by -1. Let the valid permutation be such a replacement of -1 in this sequence back to numbers from 1 to n in such a way that the resulting sequence is a permutation of size n.

The given sequence was turned into a valid permutation randomly with the equal probability of getting each valid permutation.

Calculate the expected total number of inversions in the resulting valid permutation.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \ne 0$. Report the value of $P \cdot Q^{-1} \pmod{998244353}$.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the sequence.

The second line contains n integers $p_1, p_2, \dots, p_n$ ($-1 \le p_i \le n$, $p_i \ne 0$) — the initial sequence.

It is guaranteed that all elements not equal to -1 are pairwise distinct.

Output

Print a single integer — the expected total number of inversions in the resulting valid permutation.

It can be shown that it is in the form of $\frac{P}{Q}$ where P and Q are non-negative integers and $Q \ne 0$. Report the value of $P \cdot Q^{-1} \pmod{998244353}$.

Examples

input
3 3 -1 -1
output
499122179

input
2 1 2
output
0

input
2 -1 -1
output
499122177

Note

In the first example two resulting valid permutations are possible:

- $[3, 1, 2]$ — 2 inversions;
- $[3, 2, 1]$ — 3 inversions.

The expected value is $\frac{2 \cdot 1 + 3 \cdot 1}{2} = 2.5$.

In the second example no -1 are present, thus the only valid permutation is possible — the given one. It has 0 inversions.

In the third example there are two resulting valid permutations — one with 0 inversions and one with 1 inversion.

G. Lucky Tickets

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

All bus tickets in Berland have their numbers. A number consists of n digits (n is even). Only k decimal digits $d_1, d_2, \dots, d_k$ can be used to form ticket numbers. If 0 is among these digits, then numbers may have leading zeroes. For example, if $n = 4$ and only digits 0 and 4 can be used, then 0000, 4004, 4440 are valid ticket numbers, and 0002, 00, 44443 are not.

A ticket is lucky if the sum of first $n/2$ digits is equal to the sum of remaining $n/2$ digits.

Calculate the number of different lucky tickets in Berland. Since the answer may be big, print it modulo 998244353.

Input

The first line contains two integers n and k $(2 \le n \le 2 \cdot 10^5, 1 \le k \le 10)$ — the number of digits in each ticket number, and the number of different decimal digits that may be used. n is even.

The second line contains a sequence of **pairwise distinct** integers $d_1, d_2, \dots, d_k$ $(0 \le d_i \le 9)$ — the digits that may be used in ticket numbers. The digits are given in arbitrary order.

Output

Print the number of lucky ticket numbers, taken modulo 998244353.

Examples

input
4 2 1 8
output
6

input
20 1 6
output
1

input
10 5 6 1 4 0 3
output
569725

input

```
1000 7
5 4 0 1 8 3 2
```

output

```
460571165
```

Note

In the first example there are 6 lucky ticket numbers: 1111, 1818, 1881, 8118, 8181 and 8888.

There is only one ticket number in the second example, it consists of 20 digits 6. This ticket number is lucky, so the answer is 1.