## MemSQL Start[c]UP 3.0 - Round 1

# A. Declined Finalists

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

This year, as in previous years, MemSQL is inviting the top 25 competitors from the Start[c]up qualification round to compete onsite for the final round. Not everyone who is eligible to compete onsite can afford to travel to the office, though. Initially the top 25 contestants are invited to come onsite. Each eligible contestant must either accept or decline the invitation. Whenever a contestant declines, the highest ranked contestant not yet invited is invited to take the place of the one that declined. This continues until 25 contestants have accepted invitations.

After the qualifying round completes, you know $K$ of the onsite finalists, as well as their qualifying ranks (which start at $1$, there are no ties). Determine the minimum possible number of contestants that declined the invitation to compete onsite in the final round.

### Input

The first line of input contains $K$ ($1 \le K \le 25$), the number of onsite finalists you know. The second line of input contains $r_1$, $r_2$, ..., $r_K$ ($1 \le r_i \le 10^6$), the qualifying ranks of the finalists you know. All these ranks are distinct.

### Output

Print the minimum possible number of contestants that declined the invitation to compete onsite.

**Examples**

| input |
| --- |
| 25<br>2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25 26 28 |
| **output** |
| 3 |

| input |
| --- |
| 5<br>16 23 8 15 4 |
| **output** |
| 0 |

| input |
| --- |
| 3<br>14 15 92 |
| **output** |
| 67 |

### Note

In the first example, you know all 25 onsite finalists. The contestants who ranked $1$-st, $13$-th, and $27$-th must have declined, so the answer is $3$.

# B. Lazy Security Guard

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

Your security guard friend recently got a new job at a new security company. The company requires him to patrol an area of the city encompassing exactly $N$ city blocks, but they let him choose which blocks. That is, your friend must walk the perimeter of a region whose area is exactly $N$ blocks. Your friend is quite lazy and would like your help to find the shortest possible route that meets the requirements. The city is laid out in a square grid pattern, and is large enough that for the sake of the problem it can be considered infinite.

## Input

Input will consist of a single integer $N$ ($1 \leq N \leq 10^6$), the number of city blocks that must be enclosed by the route.

## Output

Print the minimum perimeter that can be achieved.

### Examples

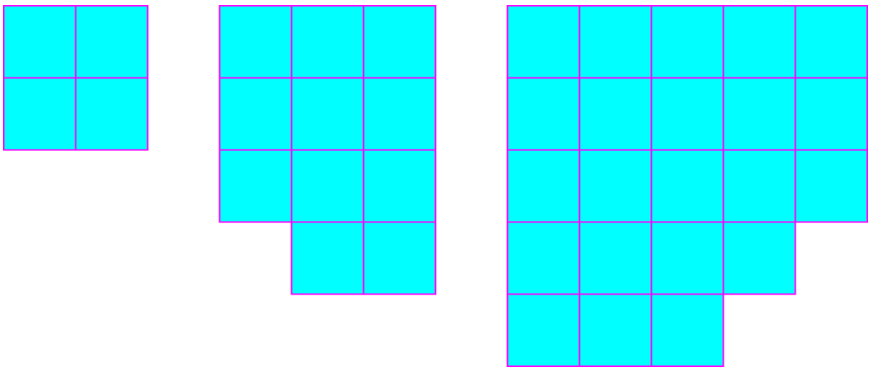
| input |
| --- |
| 4 |
| output |
| 8 |

| input |
| --- |
| 11 |
| output |
| 14 |

| input |
| --- |
| 22 |
| output |
| 20 |

## Note

Here are some possible shapes for the examples:



# C. Pie Rules

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

You may have heard of the pie rule before. It states that if two people wish to fairly share a slice of pie, one person should cut the slice in half, and the other person should choose who gets which slice. Alice and Bob have many slices of pie, and rather than cutting the slices in half, each individual slice will be eaten by just one person.

The way Alice and Bob decide who eats each slice is as follows. First, the order in which the pies are to be handed out is decided. There is a special token called the "decider" token, initially held by Bob. Until all the pie is handed out, whoever has the decider token will give the next slice of pie to one of the participants, and the decider token to the other participant. They continue until no slices of pie are left.

All of the slices are of excellent quality, so each participant obviously wants to maximize the total amount of pie they get to eat. Assuming both players make their decisions optimally, how much pie will each participant receive?

## Input

Input will begin with an integer $N$ ($1 \leq N \leq 50$), the number of slices of pie.

Following this is a line with $N$ integers indicating the sizes of the slices (each between $1$ and $100000$, inclusive), in the order in which they must be handed out.

## Output

Print two integers. First, the sum of the sizes of slices eaten by Alice, then the sum of the sizes of the slices eaten by Bob, assuming both players make their decisions optimally.

### Examples

| input |
| --- |
| 3<br>141 592 653 |
| **output** |
| 653 733 |

| input |
| --- |
| 5<br>10 21 10 21 10 |
| **output** |
| 31 41 |

## Note

In the first example, Bob takes the size $141$ slice for himself and gives the decider token to Alice. Then Alice gives the size $592$ slice to Bob and keeps the decider token for herself, so that she can then give the size $653$ slice to herself.

# D. Third Month Insanity

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

The annual college sports-ball tournament is approaching, which for trademark reasons we'll refer to as Third Month Insanity. There are a total of $2^N$ teams participating in the tournament, numbered from $1$ to $2^N$. The tournament lasts $N$ rounds, with each round eliminating half the teams. The first round consists of $2^{N-1}$ games, numbered starting from $1$. In game $i$, team $2 \cdot i - 1$ will play against team $2 \cdot i$. The loser is eliminated and the winner advances to the next round (there are no ties). Each subsequent round has half as many games as the previous round, and in game $i$ the winner of the previous round's game $2 \cdot i - 1$ will play against the winner of the previous round's game $2 \cdot i$.

Every year the office has a pool to see who can create the best bracket. A bracket is a set of winner predictions for every game. For games in the first round you may predict either team to win, but for games in later rounds the winner you predict must also be predicted as a winner in the previous round. Note that the bracket is fully constructed before any games are actually played. Correct predictions in the first round are worth $1$ point, and correct predictions in each subsequent round are worth twice as many points as the previous, so correct predictions in the final game are worth $2^{N-1}$ points.

For every pair of teams in the league, you have estimated the probability of each team winning if they play against each other. Now you want to construct a bracket with the maximum possible expected score.

## Input

Input will begin with a line containing $N$ ($2 \leq N \leq 6$).

$2^N$ lines follow, each with $2^N$ integers. The $j$-th column of the $i$-th row indicates the percentage chance that team $i$ will defeat team $j$, unless $i = j$, in which case the value will be $0$. It is guaranteed that the $i$-th column of the $j$-th row plus the $j$-th column of the $i$-th row will add to exactly $100$.

## Output

Print the maximum possible expected score over all possible brackets. Your answer must be correct to within an absolute or relative error of $10^{-9}$.

Formally, let your answer be $a$, and the jury's answer be $b$. Your answer will be considered correct, if .

**Examples**

| input |
| --- |
| 2<br>0 40 100 100<br>60 0 40 40<br>0 60 0 45<br>0 60 55 0 |

| output |
| --- |
| 1.75 |

| input |
| --- |
| 3<br>0 0 100 0 100 0 0 0<br>100 0 100 0 0 0 100 100<br>0 0 0 100 100 0 0 0<br>100 100 0 0 0 0 100 100<br>0 100 0 100 0 0 100 0<br>100 100 100 100 100 0 0 0<br>100 0 100 0 0 100 0 0<br>100 0 100 0 100 100 100 0 |

| output |
| --- |
| 12 |

| input |
| --- |
| 2<br>0 21 41 26<br>79 0 97 33<br>59 3 0 91<br>74 67 9 0 |

| output |
| --- |
| 3.141592 |

## Note

In the first example, you should predict teams $1$ and $4$ to win in round $1$, and team $1$ to win in round $2$. Recall that the winner you predict in round $2$ must also be predicted as a winner in round $1$.

# E. Desk Disorder

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

A new set of desks just arrived, and it's about time! Things were getting quite cramped in the office. You've been put in charge of creating a new seating chart for the engineers. The desks are numbered, and you sent out a survey to the engineering team asking each engineer the number of the desk they currently sit at, and the number of the desk they would like to sit at (which may be the same as their current desk). Each engineer must either remain where they sit, or move to the desired seat they indicated in the survey. No two engineers currently sit at the same desk, nor may any two engineers sit at the same desk in the new seating arrangement.

How many seating arrangements can you create that meet the specified requirements? The answer may be very large, so compute it modulo $1000000007 = 10^9 + 7$.

## Input

Input will begin with a line containing $N$ ($1 \leq N \leq 100000$), the number of engineers.

$N$ lines follow, each containing exactly two integers. The $i$-th line contains the number of the current desk of the $i$-th engineer and the number of the desk the $i$-th engineer wants to move to. Desks are numbered from $1$ to $2 \cdot N$. It is guaranteed that no two engineers sit at the same desk.

## Output

Print the number of possible assignments, modulo $1000000007 = 10^9 + 7$.

**Examples**

| input |
| --- |
| 4<br>1 5<br>5 2<br>3 7<br>7 3 |
| **output** |
| 6 |

| input |
| --- |
| 5<br>1 10<br>2 10<br>3 10<br>4 10<br>5 5 |
| **output** |
| 5 |

## Note

These are the possible assignments for the first example:

- 1 5 3 7
- 1 2 3 7
- 5 2 3 7
- 1 5 7 3
- 1 2 7 3
- 5 2 7 3

# F. Ordering T-Shirts

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

It's another Start[c]up, and that means there are T-shirts to order. In order to make sure T-shirts are shipped as soon as possible, we've decided that this year we're going to order all of the necessary T-shirts before the actual competition. The top $C$ contestants are going to be awarded T-shirts, but we obviously don't know which contestants that will be. The plan is to get the T-Shirt sizes of all contestants before the actual competition, and then order enough T-shirts so that no matter who is in the top $C$ we'll have T-shirts available in order to award them.

In order to get the T-shirt sizes of the contestants, we will send out a survey. The survey will allow contestants to either specify a single desired T-shirt size, or two adjacent T-shirt sizes. If a contestant specifies two sizes, it means that they can be awarded either size.

As you can probably tell, this plan could require ordering a lot of unnecessary T-shirts. We'd like your help to determine the minimum number of T-shirts we'll need to order to ensure that we'll be able to award T-shirts no matter the outcome of the competition.

## Input

Input will begin with two integers $N$ and $C$ ($1 \le N \le 2 \cdot 10^5$, $1 \le C$), the number of T-shirt sizes and number of T-shirts to be awarded, respectively.

Following this is a line with $2 \cdot N - 1$ integers, $s_1$ through $s_{2 \cdot N - 1}$ ($0 \le s_i \le 10^8$). For odd $i$, $s_i$ indicates the number of contestants desiring T-shirt size $((i + 1) / 2)$. For even $i$, $s_i$ indicates the number of contestants okay receiving either of T-shirt sizes $(i / 2)$ and $(i / 2 + 1)$. $C$ will not exceed the total number of contestants.

## Output

Print the minimum number of T-shirts we need to buy.

**Examples**

| input |
| --- |
| 2 200<br>100 250 100 |
| **output** |
| 200 |

| input |
| --- |
| 4 160<br>88 69 62 29 58 52 44 |
| **output** |
| 314 |

## Note

In the first example, we can buy $100$ of each size.

# G. Circle of Numbers

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

$n$ evenly spaced points have been marked around the edge of a circle. There is a number written at each point. You choose a positive real number $k$. Then you may repeatedly select a set of $2$ or more points which are evenly spaced, and either increase all numbers at points in the set by $k$ or decrease all numbers at points in the set by $k$. You would like to eventually end up with all numbers equal to $0$. Is it possible?

A set of $2$ points is considered evenly spaced if they are diametrically opposed, and a set of $3$ or more points is considered evenly spaced if they form a regular polygon.

## Input

The first line of input contains an integer $n$ ($3 \le n \le 100000$), the number of points along the circle.

The following line contains a string $s$ with exactly $n$ digits, indicating the numbers initially present at each of the points, in clockwise order.

## Output

Print "YES" (without quotes) if there is some sequence of operations that results in all numbers being $0$, otherwise "NO" (without quotes).

You can print each letter in any case (upper or lower).

### Examples

| input |
| --- |
| 30<br>000100000100000110000000001100 |
| **output** |
| YES |

| input |
| --- |
| 6<br>314159 |
| **output** |
| NO |

## Note

If we label the points from $1$ to $n$, then for the first test case we can set $k = 1$. Then we increase the numbers at points $7$ and $22$ by $1$, then decrease the numbers at points $7$, $17$, and $27$ by $1$, then decrease the numbers at points $4$, $10$, $16$, $22$, and $28$ by $1$.