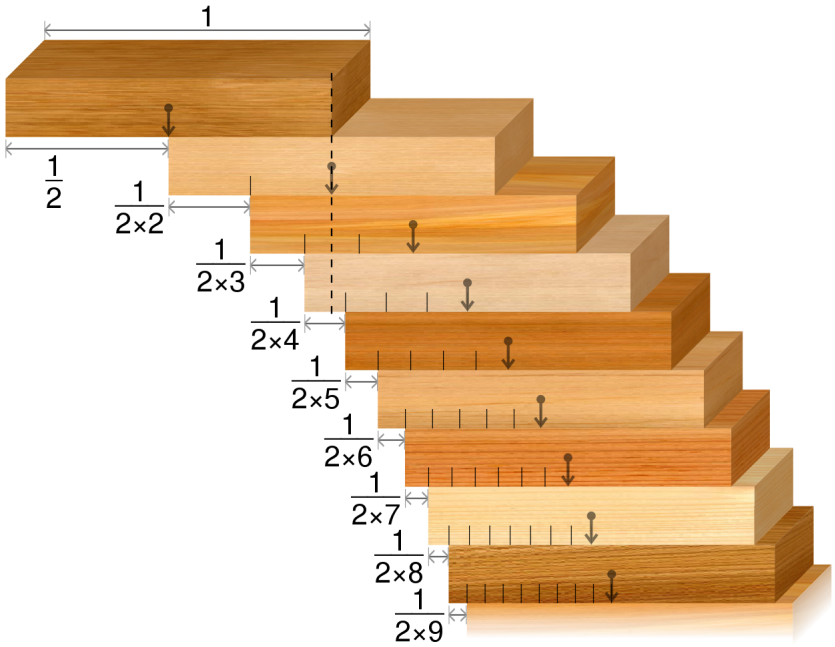
A. King of Blocks (Tower)

Problem Statement

There's a famous problem about stacking blocks in physics: "there are n identical blocks of length 1, how long can the stack be extended from the base if we stack the blocks one by one?"

In fact, the answer for this problem is exactly $\sum_{i=1}^{n} \frac{1}{2i}$, so if there's enough blocks, the stack can be extended without limit. Refer to the diagram below.



(Wikipedia, Uploaded by cmglee, Anonimski, Block stacking problem.svg, under CC BY-SA 4.0 License)

The King of Blocks thinks that this problem is a joke, so he summoned some blocks of equal length but not necessarily of equal weight, the King of Blocks wants to know how long the stack can extend in this case.

Formally, there are n uniform rectangular blocks, the i-th of which has weight w_i and length L. Each block has to be either on top of another block or the table, and each block and the table can have at most one block stacked on top. You can stack blocks in an arbitrary order. Each block must be placed such that the long side is perpendicular to the edge of the table. The table is rectangular with infinite length and width.

If the center of mass of the top x blocks is not above the $x+1$-th block for some x, these blocks will topple and fall over. In other terms, if we let the coordinate of the left side of the $x+1$-th block be 0, and the j-th block is centered at p'_j and has weight w'_j, they must satisfy

$$0 \leq \frac{\sum_{j=1}^{x} p'_j w'_j}{\sum_{j=1}^{x} w'_j} \leq L$$

Of course, the center of mass of all the blocks has to be on the table. Suppose the left side of the table has

coordinate 0. How long can the stack be extended to the left?

Your answer will be considered correct if the absolute or relative error is less than 10^{-9}.

Input Format

$n\ L$ $w_1\ w_2\ \ldots\ w_n$

- n denotes the number of blocks.
- L denotes the length of each block.
- w_i is the weight of the ith block.

Output Format

ans

- ans denotes the maximum length that the stack can be extended from the edge of the table. Your answer will be considered correct if the absolute or relative error is less than 10^{-9}.

Constraints

- $1 \leq N \leq 3 \times 10^5$
- $1 \leq L \leq 10^9$
- $1 \leq w_i \leq 10^9$
- All inputs are integers

Example

Sample Input	Sample Output
1 1 1	0.5
3 12 1 2 3	13

Scoring

There are 3 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional Constraints
1	25	$n \leq 9$
2	25	$w_i = 1, \forall 1 \leq i \leq n$
3	50	No other constraints


NHSPC
MOCK
全國資訊學科能力競賽

B. Quadrivial (Quadrivial)

Problem Statement

Kitty enjoys competitive programming, but she is currently troubled by setting new problems for NHSPC.

"How on earth do I compose a non-trivial problem?" Day after day she wondered.

For this, Kitty scoured through theses, studied new tricks, discussed algorithms with others (like quad-connected component in $O(n+m)$). Even after all that, she still couldn't compose a non-trivial problem.

In the late, dark night, a day before the deadline, with Laffey by her side, an idea struck: Why not compose a problem about problem composing?

Kitty and Laffey worked together to write down every problem they knew and every trick they studied in the form of integer sequences. In a **brainstorming session**, they permute all problems and tricks and chains them back to back to form a long sequence, after that, in a **composing session**, they pick a non-empty interval from that sequence to be their problem. The greatness of the problem is the sum of the integers in that interval. Kitty is interested in the greatest problem possible from all possible brainstorming and composing sessions.

More formally, let's say Kitty and Laffey knew n problems and tricks, the i-th of them can be written as an integer sequence $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$. Kitty first decides a permutation of 1 to n $p_1, p_2, \ldots, p_N$, and connects the sequences p_1-th problem, p_2-th problem, ..., p_n-th problem in that order. Kitty then chooses a non-empty interval from the connected sequences and computes the sum of the integers in that interval. She is interested to know the largest possible sum that can be obtained.

Smart as Kitty is, she noticed instantly that connecting problems and tricks doesn't make a problem non-trivial, instead, it's even more trivial! She didn't have time to compose another problem so this is the one she submitted. As the problem is more trivial than ever, she decides to call it "Quadrivial".

Input Format

$$
\begin{array}{l}
n \\
k_1 \ a_{1,1} \ a_{1,2} \ \cdots \ a_{1,k_1} \\
k_2 \ a_{2,1} \ a_{2,2} \ \cdots \ a_{2,k_2} \\
\vdots \\
k_n \ a_{n,1} \ a_{n,2} \ \cdots \ a_{n,k_n}
\end{array}
$$

- n denoted the number of integer sequences.
- k_i denotes the size of the i-th sequence.
- $a_{i,j}$ denotes the j-th element in the i-th sequence($1 \le i \le n$, $1 \le j \le k_i$).

Output Format

ans

- ans denotes the greatness of greatest problem possible.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq k_i \leq 10^5 (1 \leq i \leq n)$
- The sum of lengths of the n sequences, $\sum_{i=1}^{n} k_i \leq 10^6$
- $|a_{i,j}| \leq 10^9 (1 \leq i \leq n \text{、} 1 \leq j \leq k_i)$
- All inputs are integers

Example

Sample Input	Sample Output
4 3 1 4 -6 2 -1 -3 6 6 -11 5 -13 3 12 4 14 -7 -3 15	39
1 1 -1000000000	-1000000000

Scoring

There are 4 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	8	$n = 1$
2	14	$n \leq 400$
3	22	$n \leq 3000$
4	56	No other constraints

C. Ceremonial Music (Ceremony)

Problem Statement

After many years of hardwork, you finally won first place in NHSPC! Still feeling excited from the adrenaline rush, you happened to walk past the award ceremony venue just before the ceremony, hearing the sound check for the music on the ceremony.

"This sounds horrible..." You don't think the music is well suited for the ceremony. Fortunately, as a master of algorithms, a musical genius and an IT security guru, you decided to take matters into your own hands, and spice up the ceremonial music.

The ceremonial music is composed of N notes, with its i-th note having pitch a_i. You can modify the music by changing the order of the notes, but in order to not get caught, in one operation, you can only choose an interval with the same starting pitch and ending pitch, then reverse the entire interval.

Formally, in each operation, you choose two indices $1 \leq l \leq r \leq N$ With $a_l = a_r$, and reverse the interval from $a_l, a_{l+1}, \ldots, a_r$ to $a_r, a_{r-1}, \ldots, a_l$. You can perform this operation arbitrarily many (possibly 0) times.

You think melodies of the form "a low pitch, then a high pitch, then a low pitch" and "a high pitch, then a low pitch, then a high pitch" are bad. So you want to minimize the number of such melodies. In other words, after the operations are done, you want the number of indices $i (1 < i < N)$ satisfying $a_{i-1} < a_i > a_{i+1}$ or $a_{i-1} > a_i < a_{i+1}$ to be the lowest possible.

Please write a program that finds the optimal list of notes after performing any number of operations.

Input Format

N
$a_1 \ a_2 \ \ldots \ a_N$

- N denotes the number of notes in the ceremonial music.
- a_i is the original pitch of the i-th note in the music.

Output Format

$b_1 \ b_2 \ \ldots \ b_N$

- b_i is the pitch of the i-th note after all of the operations has been done.
- If there's more than one answer with minimal bad melodies, any one of them will be accepted.

Constraints

- $3 \leq N \leq 10^6$
- $1 \leq a_i \leq N$
- All inputs are integers

Example

Sample Input	Sample Output
10 2 2 6 2 1 3 6 4 1 4	2 6 4 1 2 2 6 3 1 4
10 2 5 5 5 10 2 3 3 10 2	2 10 5 5 5 2 3 3 10 2
10 8 4 9 4 6 5 8 2 4 8	8 2 4 9 4 6 5 8 4 8
7 2 3 2 1 4 3 4	2 3 4 3 2 1 4

Scoring

There are 5 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	9	$N \leq 10$
2	20	$a_i \leq 2$
3	17	$N \leq 1000$
4	21	$N \leq 10^5$
5	33	No other constraints

D. Dr. Tornado's Game (Game)

Problem Statement

Dr. Tornado was asked to design a game by NHSPCCamp, the game consists of N cities and M bi-directional roads connecting the cities. The cities are indexed from 1 to N. Initially the player is at city S, the goal of the player is to get to T via a series of roads.

There's a monster on each road, and their health is an integer no greater than 10^{12}, if the player wants to go through a road, the monster on that road must be killed. The monster respawns once the road is crossed. For a monster of health x, the player needs exactly x seconds to kill it, we can treat the health of the monster to be the time it takes to cross the road.

Dr. Tornado doesn't want his game to be beaten too quickly, nor does he want it to be so dull that no one wants to play it, so he wants the minimum possible time to go from S to T be exactly D seconds.

Dr. Tornado has planned out the relative strength of the monsters. In formal terms, suppose the i-th road connects u_i, v_i, and the health of the monster on that road has health h_i. Dr. Tornado has a permutation of 1 to M, denoted by $p_1, p_2, \ldots, p_M$, meaning the health of the monsters should satisfy $h_{p_1} < h_{p_2} < \cdots < h_{p_M}$.

He'll be busy for a while, so he can't work on the game, fortunately the only thing that hasn't been decided yet is the health of the monsters. Given the known information, is it possible to construct a game that satisfies his requirements? If so, please construct such a game for him.

Input Format

$N\ M\ S\ T\ D$
$p_1\ p_2\ \cdots\ p_M$
$u_1\ v_1$
$u_2\ v_2$
$\vdots$
$u_M\ v_M$

- N, M denotes the number of cities and roads, respectively.
- S, T denotes the index of the starting and ending city, respectively.
- D is the amount of time mentioned above.
- p_i means h_{p_i} is the i-th smallest health of all monsters.
- u_i, v_i denotes the endpoints of the i-th road.

Output Format

ans $h_1\ h_2\ \ldots\ h_M$

- ans is YES or NO, if there exists a valid sequence of monster health $h_1, h_2, \ldots, h_M$ then ans is YES, otherwise it's NO.
- the second line is needed only when ans is YES.
- $1 \leq h_i \leq 10^{12}$.
- h_i is a positive integer.
- $h_1, h_2, \ldots, h_M$ is a sequence of monster health satisfying all constraints mentioned above, if there's more than one solution, any one of them will be accepted. If your code fails to output a valid sequence, you would obtain 50% of the score of that subtask.

Constraints

- $2 \leq N \leq 10^5$
- $N - 1 \leq M \leq \min\{2 \times 10^5, \frac{N(N-1)}{2}\}$
- $1 \leq S, T \leq N$
- $S \neq T$
- $1 \leq D \leq 10^{11}$
- $1 \leq p_i \leq M$
- $1 \leq u_i, v_i \leq N$
- $u_i \neq v_i$
- No two roads connect the exact same cities
- Any city can go to every other city through a series of roads
- $p_1, p_2, \ldots, p_M$ are pairwise distinct
- All inputs are integers

Example

Sample Input	Sample Output
4 5 1 4 10 5 1 4 3 2 1 3 1 2 2 3 3 4 2 4	YES 4 8 7 6 3
5 4 3 4 1 1 2 3 4 3 1 1 2 2 5 5 4	NO

Scoring

There are 4 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	5	$M = N - 1$
2	9	$M = N$, and every city is connected to exactly two roads
3	23	$N \leq 3000, M \leq 5000$
4	63	No other constraints

Additionally, if your code correctly determines whether a valid sequence exists, but fails to provide such a valid sequence, you would get 50% of the score of that subtask. Note that you should still output a sequence of M numbers in the range $[1, 10^{12}]$ when the answer is YES.


NHSPC
MOCK
全國資訊學科能力競賽

E. Escape the Common Room (Escape)

Problem Statement

The CSIE(Computer Science and Informatic Engineering) common room of National Okapi University is a strange place. There's a lounge for each year (freshman, sophomore, etc.), a Mahjong room, a study hall and more, but there also resides a monster called the Three-mouthed Sheep.

Three-mouthed sheep is a terrifying creature that likes to (physically) merge with freshmen. When an unsuspecting target is spotted, it will start approching them, and they will definitely get merged by this monster once they run into each other.

Recently something unimaginable happened, the Three-mouthed Sheep started multiplying! More and more Three-mouthed sheep started to appear out of nowhere, with the sole purpose of traumatizing freshmen for the rest of their life! **8e7**, an innocent freshman, wants to avoid being merged by the Three-mouthed sheep.

Formally, the common room is a $C \times C$ grid in an infinite plane, there are N three-mouthed sheep which will appear in order, the i-th of which will appear at integer coordinates (x_i, y_i). It is guaranteed that $0 \leq x_i, y_i \leq C$. Initially **8e7** starts at integer coordinates (x, y) in the $C \times C$ grid, and all of the three-mouthed sheep will notice **8e7** and try to merge with him. He is considered merged if at any point in time his position coincides with any of the three-mouthed sheep (including the initial position). In each move, **8e7** can move one unit along the grid (facing up, down, left, or right) or stay still, then all of the three-mouthed sheep will also move one unit along the grid or stay still. Each three-mouthed sheep moves independently, and they can move optimally as a group to catch **8e7**.

A starting position (x, y) is considered safe if **8e7** can escape the $C \times C$ area after some moves, and dangerous otherwise.

For each i, **8e7** wants to know how many dangerous starting positions there are after the first i three-mouthed sheep appear.

Please write a program to help **8e7** out (He really needs it).

Input Format

n C
x_1 y_1
x_2 y_2
$\vdots$
x_n y_n

- n denotes the number of three-mouthed sheep.
- C denotes the side length of the common room.
- x_i, y_i denotes the starting position of the i-th three-mouthed sheep.

Output Format

ans_1
ans_2
$\vdots$
ans_n

- ans_i denotes the numder of dangerous starting position after the first i three-mouthed sheep appear.

Constraints

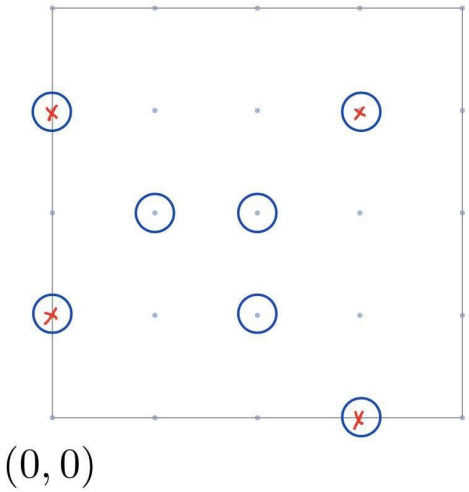
- $1 \leq N, C \leq 2 \times 10^5$
- $0 \leq x_i, y_i \leq C, \forall 1 \leq i \leq N$
- All inputs are integers

Example

Sample Input	Sample Output
4 4 0 3 0 1 3 0 3 3	1 2 5 7
5 7 0 0 1 0 1 5 1 6 3 2	1 2 3 4 8

Explanation of sample test

The following diagram is an illustration of Sample Test 1 after all three-mouthed-sheep have appeared:



The red 'X's indicates the location of a three-mouthed-sheep, and the blue circles indicates a dangerous starting position.

Scoring

There are 5 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	7	$n, C \leq 30$
2	10	$n \leq 5 \times 10^4, C \leq 1000$
3	14	$n \leq 1000$
4	31	$n \leq 5 \times 10^4$
5	38	No other constraints


NHSPC
MOCK
全國資訊學科能力競賽

F. LCA Game (LCA)

Problem Statement

Alice and Bob are two very bored siblings. They often play games on various data structures and graphs that are way too big. One day, Alice gave Bob a tree with N vertices, labeled from 1 to N, and marked a vertex on the tree as A. Then, Alice chose a vertex B on the tree, keeping it secret. Bob's goal is to find vertex B.

In one move, Bob chooses a vertex C on the tree, and Alice would tell Bob the LCA (lowest common ancestor) of A and B if the tree was rooted at C.

Alice wants to maximize the number of guesses that Bob needs, so she decides to cheat a little bit. As Bob is guessing, Alice can change her secret vertex B, as long as all of her previous answers are consistent with the new vertex she changes to.

Knowing Alice, Bob decided to employ a strategy that would minimize his guesses. Please help Bob compute the minimum number of guesses he needs to find B regardless of Alice's choices.

Input Format

N A
u_1 v_1
u_2 v_2
$\vdots$
u_{N-1} v_{N-1}

- N denotes the number of vertices.
- A denotes the index of vertex A.
- u_i, v_i is the endpoints of the i-th edge.

Output Format

ans

- ans is the maximal number of times Bob needs to find out vertex B.

Constraints

- $1 \leq N \leq 10^6$
- $1 \leq u_i, v_i, A \leq N$

- The given graph is a tree
- All inputs are integers

Example

Sample Input	Sample Output
5 1 1 2 2 3 3 4 4 5	1
5 3 1 2 2 3 3 4 4 5	2
7 1 1 2 1 3 2 4 2 5 3 6 3 7	3

Scoring

There are 2 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	29	There exists a vertex v, where the input forms a binary tree when rooted at v
2	71	No other constraints

G. Lucky Number (Number)

Problem Statement

BB is a genius that loves travelling, recently he decided to travel around the world despite his girlfriend opposing him!

BB decides to go to Q very special countries. For the i-th country, they only issue bills of denomination $1, b_i, b_i^2, b_i^3, \ldots$ (i.e. non-negative powers of b_i). There are N souvenirs at every country, the i-th souvenir costs a_i dollars in that country's currency, for every country from 1 to Q.

For some reason, BB is obsessed with the integer M, so he defines the "greatness" of a country as the number of souvenirs in that country such that M is the minimal number of bills required to buy that souvenir.

BB only likes to travel, so he asked you to find the greatness of every country he wants to go, that way he can plan out the rest of the trip, can you write a program solving BB's problem?

Input Format

$N\ M\ Q$
$a_1\ a_2\ \ldots\ a_n$
b_1
b_2
$\ldots$
b_Q

- The meaning of N, M, Q, a_i, b_i is described in the statement.

Output Format

ans_1
ans_2
$\ldots$
ans_Q

- ans_i is the greatness of the i-th country.

Constraints

- $1 \leq N \leq 2 \times 10^6$
- $1 \leq M, Q, a_i \leq 2 \times 10^6$

- $2 \leq b_i \leq 10^9$
- All inputs are integers

Example

Sample Input	Sample Output
10 3 10 1 2 3 4 5 6 7 8 9 10 2 3 4 5 6 7 8 9 10 11	1 2 3 2 2 2 2 1 1 1

Scoring

There are 4 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	9	$N, Q \leq 1000$
2	29	$a_i \leq 40000$
3	24	$a_i \leq 3 \times 10^5$
4	38	No other constraints

H. Compass (Compass)

Problem Statement

Kiwi is good at solving problems about circles. For example, intersection of circles, union of circles, and union of circular sectors all failed to stump him.

Today, **Kiwi** stumbled upon a grid of size 7001×7001, the center of the grid is $(0, 0)$. Each cell in the grid could be colored either black or white. Initially, every cell is white except the cell at $(0, r)$.

Due to the size of the grid, **Kiwi** can only observe it from the outside, so **Kiwi** doesn't know the value of r, however he's sure that $0 < r \leq 3000$.

Kiwi wants to draw a circle with center $(0, 0)$ and radius r. You might be wondering how to draw a circle on a grid, so **Kiwi** shared his definition of a circle.

- For $0 \leq x \leq \sqrt{r^2 - x^2}$, cell $(x, \lfloor\sqrt{r^2 - x^2}\rfloor)$, $(-x, \lfloor\sqrt{r^2 - x^2}\rfloor)$, $(x, -\lfloor\sqrt{r^2 - x^2}\rfloor)$, $(-x, -\lfloor\sqrt{r^2 - x^2}\rfloor)$ is black.
- For $0 \leq y \leq \sqrt{r^2 - y^2}$, cell $(\lfloor\sqrt{r^2 - y^2}\rfloor, y)$, $(\lfloor\sqrt{r^2 - y^2}\rfloor, -y)$, $(-\lfloor\sqrt{r^2 - y^2}\rfloor, y)$, $(-\lfloor\sqrt{r^2 - y^2}\rfloor, -y)$ is black.
- All other cells are white.

Kiwi also stumbled upon a magical machine to aid him on this project. This machine can do the following operations:

- `set_col(x, y, len, val)`:
  $val \in \{0, 1\}$, For $0 \leq k < len$, if $val = 1$, color $(x, y + k)$ black, otherwise color it white.
- `set_row(x, y, len, val)`:
  $val \in \{0, 1\}$, For $0 \leq k < len$, if $val = 1$, color $(x + k, y)$ black, otherwise color it white.
- `not_col(x, y, xout, yout, len)`:
  For $0 \leq k < len$, if $(x, y+k)$ is white originally, color $(x_{out}, y_{out}+k)$ black, otherwise color it white.
- `not_row(x, y, xout, yout, len)`:
  For $0 \leq k < len$, if $(x+k, y)$ is white originally, color $(x_{out}+k, y_{out})$ black, otherwise color it white.
- `and_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \leq k < len$, if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **are both black originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `and_row(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \leq k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **are both black originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.
- `or_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \leq k < len$, if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **has at least one black cell originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `or_row(x1, y1, x2, y2, xout, yout, len)`:

For $0 \leq k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **has at least one black cell originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.

- `xor_col(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \leq k < len$,if $(x_1, y_1 + k)$ and $(x_2, y_2 + k)$ **has exactly one black cell originally**, color $(x_{out}, y_{out} + k)$ black, otherwise color it white.
- `xor_row(x1, y1, x2, y2, xout, yout, len)`:
  For $0 \leq k < len$, if $(x_1 + k, y_1)$ and $(x_2 + k, y_2)$ **has exactly one black cell originally**, color $(x_{out} + k, y_{out})$ black, otherwise color it white.

The machine reads the color of required cell for each operation first, then modify all the cell at once, that is, if the output overlaps with the input, the original state of the input will be read.

Please note that if the operation **Kiwi** performs causes the machine to read or write to cells outside of the grid, the machine explodes and so does **Kiwi**. If more than 6×10^6 operations are done, the machine also explodes. Although this machine is full of danger, **Kiwi** really wants to draw circles. Having said that, **Kiwi** is tired, so he asked you to tell him how to operate the machine so that regardless of the value of r, the machine draws the perfect circle that he has in mind.

Implementation Details

You must add `#include "Compass.h"` in the first line of your code, and implement the following function.

```
void draw_circle();
```

Your program can call the following functions.

```
void set_col(int x, int y, int len, int val);
void set_row(int x, int y, int len, int val);
void not_col(int x, int y, int xout, int yout, int len);
void not_row(int x, int y, int xout, int yout, int len);
void and_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void and_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void or_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void or_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void xor_col(int x1, int y1, int x2, int y2, int xout, int yout, int len);
void xor_row(int x1, int y1, int x2, int y2, int xout, int yout, int len);

void debug(int x, int y);
```

- `debug(x, y)` askes the sample grader to output the color of cell (x, y), where 1 is black and 0 is white. in the actual grading process, this function does nothing and does not count as an operation.
- the effect of other functions is described in the statement.
- `val` must be `0` or `1`.
- `x,y,x1,y1,x2,y2,xout,yout` must be integers between $[-3500, 3500]$.
- `len` must be a positive integer.

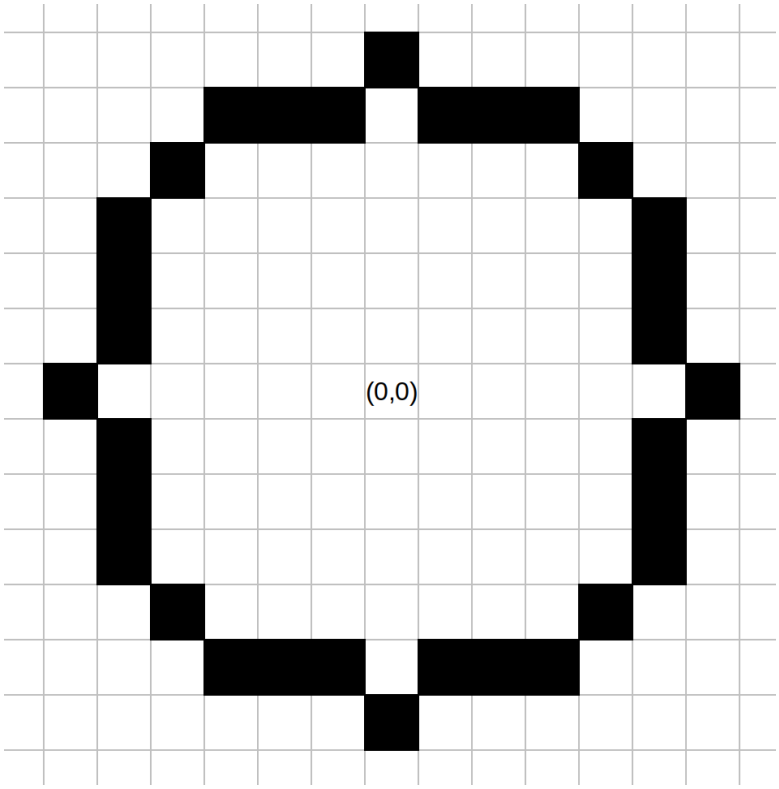
- For `not_col,set_col,and_col,or_col,xor_col`, adding `len-1` to `y,y1,y2,yout` must still be within $[-3500, 3500]$.
- For `not_row,set_row,and_row,or_row,xor_row`, adding `len-1` to `x,x1,x2,xout` must still be within $[-3500, 3500]$.
- You can make at most 6×10^6 calls to functions other than `debug()`.
- The sum of `len` for every call must not exceed 10^9.

Sample Interaction

When $r = 6$, here's a sample program that draws the correct circle:

```
set_col(5, 1, 3, 1);
set_col(5, -3, 3, 1);
or_row(6, 0, 5, 0, 4, 4, 1);
set_row(-3, 5, 7, 1);
and_col(0, 4, 0, 4, 0, 5, 2);
not_row(-4, 4, -4, 4, 1);
xor_col(-5, -3, 5, -3, -5, -3, 7);
xor_col(-5, 0, 6, 0, -6, 0, 1);
set_row(-3, -5, 7, 1);
not_col(0, -6, 0, -6, 2);
set_col(-4, -4, 1, 1);
set_col(4, -4, 1, 1);
```

The end result looks like:



Scoring

There are 3 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	5	$r \leq 1000$
2	10	$r \leq 2500$
3	85	No other constraints

For the third subtask, if the answer is correct, the machine didn't explode and q operations are done by **Kiwi**, your score for this subtask would be:

- If $q \leq 7.5 \times 10^5$, your score for this subtask is 85.
- If $7.5 \times 10^5 < q \leq 3.5 \times 10^6$, your score for this subtask is $85 - \frac{55}{2.75 \times 10^6}(q - 7.5 \times 10^5)$.
- If $3.5 \times 10^6 < q \leq 6 \times 10^6$, your score for this subtask is $30 - \frac{30}{2.5 \times 10^6}(q - 3.5 \times 10^6)$.

Sample Grader

Sample grader reads input with the following format:

- First line: r.

r denotes the radius of the circle.

If your program is judged as Accepted, the sample grader outputs Accepted: q, where q denotes the number of calls to functions other than `debug()`. If your program is judged as Wrong Answer, the sample grader outputs Wrong Answer: MSG, the meaning of MSG is as follows:

- invalid operation: the parameter to function call is invalid, such as going out of bounds when reading or writing.
- too many operations: too many function calls, or the sum of len breaks the constraint.
- wrong result: the final grid isn't what **Kiwi** has in mind.

In the attachment, there's a compressed file named "Compass.zip", when decompressed, there are three folders: `cpp`, `c`, and `examples`. The files in these folders are as follows:

- `cpp`: There is a sample `cpp` file named `Compass.cpp`, you can modify this code or write your own, then use `compile_cpp.sh` or `compile_cpp.bat` to compile on your local computer.
- `c`: There is a sample `c` file named `Compass.c`, you can modify this code or write your own, then use `compile_c.sh` or `compile_c.bat` to compile on your local computer.
- `examples`: the input for sample interaction.

Please do not try to code anything beyond the required function, such as input, output, etc. `grader.cpp` **is only for reference, and might be different from the actual grader used by the judge.**


NHSPC
MOCK
全國資訊學科能力競賽

I. Sum Over Subsets (SOS)

Problem Statement

Sum Over Subsets is a well-known dynamic programming technique. It is also known as multi-dimensional prefix-sum in China. Given an initial set and a value for each subset denoted as $c(S)$, this technique finds the sum of values for all subsets of S for all possible subsets S.

However, this problem has nothing to do with the famous technique.

Our beloved **Ji Kuai** has a function $f(S) = |S| \times \prod_{x \in S} x$. Specially, for empty set $\varnothing$, $f(\varnothing) = 0$.

One day, **Joy** gives a multiset A as a gift to **Ji**. Since **Ji** is a mathematician, he decided to compute the sum over all subsets of A, but quickly realized that this is too trivial. As a talented competetive programmer, he thinks that the ordinary sum is too boring and wants to calculate the sum of $f(S)$ where $S \subseteq A$ in order to impress **Joy**. Since the answer could be quite large, he simply wants to find the answer modulo 998244353.

Ji solved the problem in nanoseconds and you have always envied **Ji**, so you want to solve it to be as handsome as **Ji**!

Formally, given a multiset A, find $ans = \sum_{S \subseteq A} f(S) \mod 998244353$ where f is defined above.

Input Format

N
$A_1\ A_2\ \ldots\ A_N$

- N denotes the size of the multiset.
- $A_1, A_2, \ldots A_N$ are elements of the multiset.

Output Format

ans

- ans is described in the problem description.

Constraints

- $1 \leq N \leq 10^6$
- $\forall i, 1 \leq A_i \leq 10^9$
- All inputs are integers

Example

Sample Input	Sample Output
2 2 3	17
4 1 2 3 4	326

Scoring

There are 5 subtasks in this problem. The score and additional constraints of each subtask are as follows:

Subtask	Score	Additional constraints
1	5	$N \leq 20$
2	15	$N \leq 1000$
3	30	$N \leq 10^5$
4	15	$A_i \leq 20$
5	35	No other constraints