The International Collegiate Programming Contest
Sponsored by ICPC Foundation

ACPC
AFRICA & ARAB
Collegiate Programming
Championship

Africa and Arab Collegiate Programming Championship 2021

(Contest Problems)

Luxor, Egypt
December 2021

Problem A. ACPC Sites

Input file:	`sites.in`
Output file:	`standard output`
Time limit:	`3 Seconds`
Balloon Color:	`Purple`

As the ACPC region is continuously expanding, and new countries are joining every year, multiple contest sites are taking place at the same time. The operations team, one of the very hard-working teams, are trying to model the contest sites as nodes in a graph, to find the mid-point between each two nodes (i.e. contest sites).

There are N contest sites taking places now, numbered from 1 to N and initially not connected. Then, you are given Q queries of 2 types:

- $1\ U\ V$ – add a road (of length 1) between contest site U and contest site V if there is no **path** between them (i.e. you cannot reach one of them from the other one). If the road is added, it's id becomes its order among added edges (where the first edge added takes id 1 and the next takes 2 and so on).
- $2\ U\ V$ – get the contest site/road that is at half the distance between contest site U and contest site V if there is a path between them. If there is an even number of roads between U and V, then a contest site is at half the distance, otherwise, it is a road.

Input

The first line of the input is the number of test cases T.

In each test case:

- The first line has two integers N and Q, $1 \leq N, Q \leq 3 * 10^5$ – the number of contest sites and queries, respectively.
- Q lines will follow with type $1\ U\ V$ or $2\ U\ V$, where $1 \leq U, V \leq N$ and $U \neq V$.

It is guaranteed that the sum of all N in the test cases will not exceed $3 * 10^5$, as well as the sum of all Q in the test cases.

Output

For each query, print a single line:

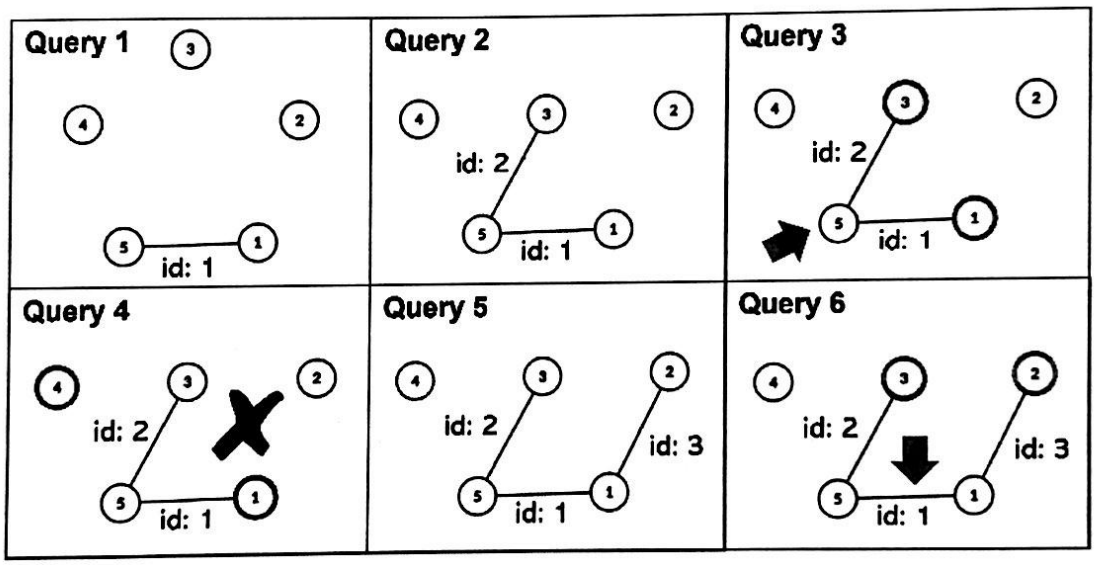
- For queries of type 1:
  - If the road is added, print the "road id".
  - Else, print -1.
- For queries of type 2:
  - If there is a contest site in the mid distance, print the letter "c", followed by a space, followed by the contest site id.
  - If there is a road in the mid distance, print the letter "r", followed by a space, followed by the road id.
  - Else, print -1.

Example

sites.in	standard output
1 5 6 1 1 5 1 3 5 2 1 3 2 1 4 1 1 2 2 2 3	1 2 c 5 -1 3 r 1

Note

A simulation of the queries in the sample test case is shown below:



Problem B. Brimore and the Carpet Market

Input file:	`brimore.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`Green`

Brimore is the biggest social commerce app in Egypt and Africa. It is a platform where businesses open their own virtual shops, share thousands of products, and enjoy instant profits.

This year, a virtual carpets shop opened on Brimore. That virtual shop sells carpets that have geometric drawings on them. While shopping for a new carpet, one of Brimore's employees saw a carpet, and was lost in the lines drawn on it.

The carpet can be modelled as a $2-D$ plane, where N straight lines are drawn on it. Brimore's employee wonders: how many formed parallelograms are on the carpet?

A parallelogram is a $2-D$ shape with 4 sides, in which every two corresponding sides are parallel.

The equation of a straight line is given by: $Y = M \cdot X + C$, where M is the slope and C is the Y-intercept.

You may assume that lines that have the same slope are guaranteed to have different Y-intercepts (i.e. No 2 lines coincide) and that there are no vertical lines.

Input

The first line of input contains one integer N – the number of lines on the carpet, where $1 \leq N \leq 10^5$.

Every line of the next N lines contains an integer M_i, which is the slope of the i^{th} line, $-10^9 \leq M_i \leq 10^9$.

Output

Output a single integer P, which is the number of formed parallelograms.

Example

brimore.in	standard output
5 500 45 500 45 45	3

Problem C. Coach Academy Tic-Tac-Toe

Input file:	`tictactoe.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`White`

Coach Academy teaches students how to think, create and code. One day, the CEO of Coach Academy wanted to play a game with the ACPC Chief Judge, so they decided to play a modified version of Tic-Tac-Toe.

Remember the normal Tic-Tac-Toe? In the normal version, your board is a rectangular 3×3 grid. Where the game goes as follows:

- The first player is denoted by the letter X.
- The second player is denoted by the letter O.
- At first the board is empty.
- The first player starts by putting X anywhere on the board.
- The second player puts O on any empty cell on the board.
- After that, they keep alternating, putting Xs and Os on empty cells.
- The first player who manages to put their letter on 3 consecutive (vertical, horizontal, or diagonal) cells, wins the game.

However, on the modified version, your board is again a 3×3 rectangular grid. **But, every cell contains a full normal Tic-Tac-Toe board.** Where the game goes as follows:

- The first player starts by putting their X anywhere they want (They choose any one of the 9 boards, and puts their X anywhere on that board).
- The second player now is restricted to play in a certain board. That board is determined by the place where first player put their X on the board they played in. For example, if the first player puts their X on the **top-left cell of the middle board**. Then, the second player is forced to put their O somewhere in the **top-left board**.
- Then, we get back to the first player and players alternate moving, where every player is restricted to play in a board determined by where the previous player decided to play in their selected board.
- In other words, the location one player plays, decides the board where the next player will play.
- The first player who manages to put their letter on 3 consecutive (vertical, horizontal, or diagonal) cells in a board, wins that board.
- What happens if a player is forced to play in an already won board?
  - If that board has empty cells, then that player has to play in one of those cells, despite that they cannot win that board over.
  - If that board is full, then they are free to play on any empty cell, on any board they choose.

You will be given the moves each player played in order, and you are asked to output who won each of the 9 boards, or to say that these moves are invalid. In case of invalid moves, you should output the first invalid move that was played.

Input

The first line of the input will be an integer N ($1 \leq N \leq 81$), the number of moves played in the game.

N lines follow, each line has exactly 2 space separated strings, the first one is the location of the board the player in turn played in, and the second string is the location of the cell in that board the player in turn put their letter.

Each string can be one of the following:

- "TL" means top left cell / board.
- "TM" means top middle cell / board.
- "TR" means top right cell / board.
- "ML" means middle left cell / board.
- "M" means middle cell / board.
- "MR" means middle right cell / board.
- "BL" means bottom left cell / board.
- "BM" means bottom middle cell / board.
- "BR" means bottom right cell / board.

Output

If one of the moves was invalid, print one line containing the word "INVALID" followed by a space, then followed by the first invalid move you found.

If all moves were valid, draw the Tic-Tac-Toe main board, where you write "X" in every cell denoting a board won by the first player, "O" in every cell denoting a board won by the second player, and "E" in every cell denoting a cell that was not won by either player.

Drawing should be as follows:

- First line should contain the letters of the top 3 boards separated by the character '|'.
- Second line should contain the letter '-' 5 times.
- Third line should contain the letters of the middle 3 boards separated by the character '|'.
- Fourth line should contain the letter '-' 5 times.
- Fifth line should contain the letters of the bottom 3 boards separated by the character '|'.

Examples

tictactoe.in	standard output
20 M M M TL TL M M TM TM M M TR TR M M MR MR M M BR BR M M BM BM M M BL BL M M ML ML M TR TR TR BL TL TL	INVALID TL TL
22 M TL TL M M TM TM M M TR TR TR TR TL TL TL TL TM TM TM TM TL TL BR BR TR TR MR MR TR TR BR BR M M BR BR BL BL ML ML TM TM BM	O|O|O ----- E|X|E ----- E|E|X

Problem D. Dominos in the Chill Zone

Input file:	`dominos.in`
Output file:	`standard output`
Time limit:	`3 Seconds`
Balloon Color:	`Red`

This year, the ACPC decided to have a Chill Zone, where any ACPC attendee can chill, play games, and socialize. The Chill Zone is modelled as a rectangular grid, and the ACPC wants to put the maximum number of games in the Chill Zone. Every game will be played on a table, and a table is a giant 1×2 domino piece that occupies 2 adjacent cells. However, you cannot place those tables anywhere, because some of the cells in the Chill Zone might be blocked.

You are given an $N \times M$ empty grid (the Chill Zone), and your task is to process two types of queries:

- $1\ X\ Y$ – block the primary diagonal that starts with cell (X, Y). In other words: block all cells $\{(X, Y), (X+1, Y+1), (X+2, Y+2), ..etc.\ \}$ as long as $X \leq N$ and $Y \leq M$. It's guaranteed that $X = 1$ **or** $Y = 1$.
- $2\ X\ Y$ – block the secondary diagonal that starts with cell (X, Y), In other words: block all cells $\{(X, Y), (X+1, Y-1), (X+2, Y-2), ..etc.\ \}$ as long as $X \leq N$ and $Y \geq 1$. It's guaranteed that $X = 1$ **or** $Y = M$.

After each query, you should print the maximum number of tables (1×2 dominos) you can put in the remaining unblocked cells.

Input

The first line contains a single integer T – the number of test cases.

For each test case:

- The first line contains two integers N and M ($1 \leq N, M \leq 2\ 000$) – the dimensions of the Chill Zone.
- The second line contains a single integer Q, ($1 \leq Q \leq 2 \times (N + M - 1)$) – the number of queries.
- Each line of the following Q lines contains three integers: $type, X, Y$, where $1 \leq type \leq 2$, $1 \leq X \leq N$ and $1 \leq Y \leq M$.

It's guaranteed that the sum of $N \times M$ over all test cases will not exceed 4×10^6.

Output

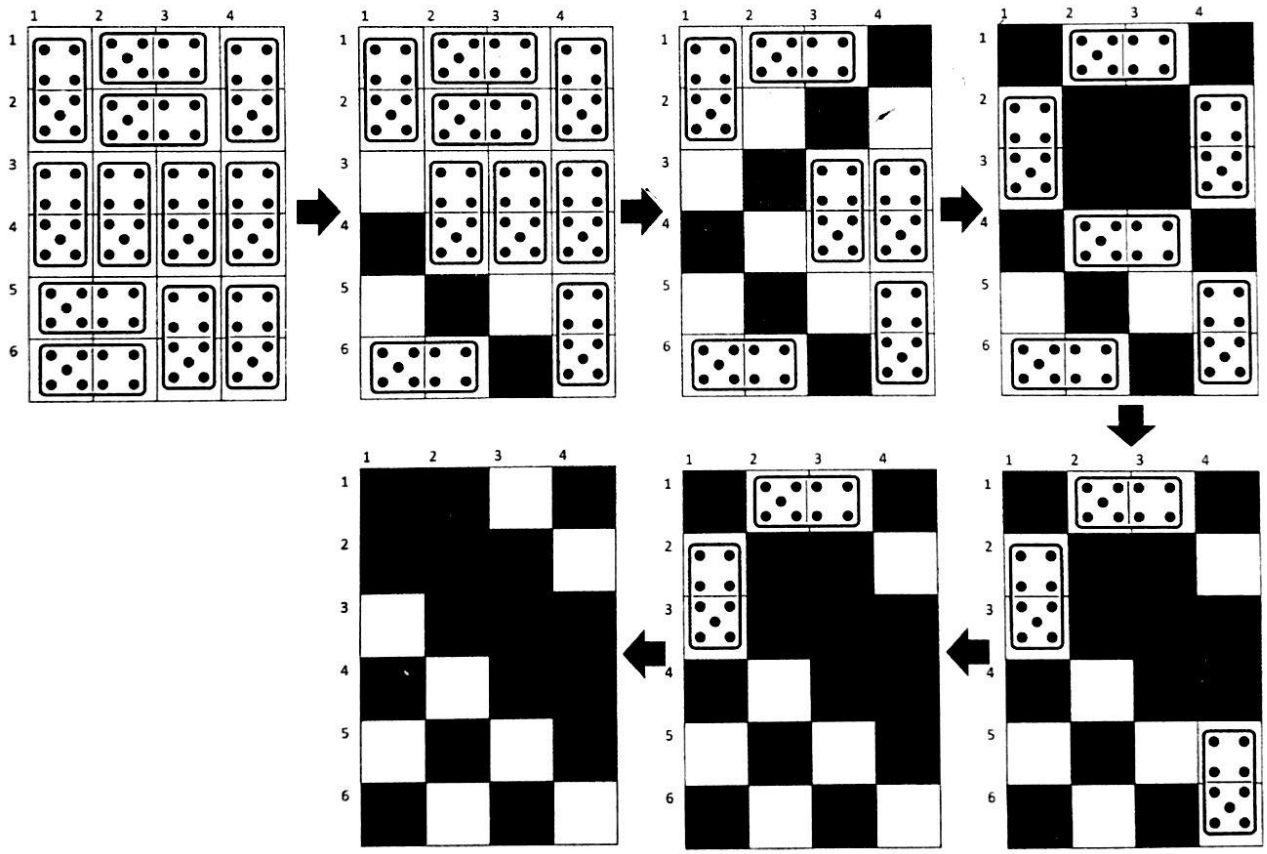
For each test case, and for each query in this test case, print a single integer that denotes the maximum number of tables (dominos) that can be placed in the Chill Zone at this moment.

Example

dominos.in	standard output
2 6 7 4 1 3 1 1 1 3 1 1 6 2 3 7 6 4 6 1 4 1 2 1 4 1 1 1 2 3 4 2 5 4 2 1 2	17 12 11 9 9 6 6 3 2 0

Note

An illustration of the queries in the second test case is:



Problem E. Endure Invests in Apples

Input file:	`apples.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`Orange`

Endure Capital is an early stage investment fund headed by entrepreneurs. Unlike many venture capital firms, Endure prides itself on being founder-focused. It believes in the people behind the idea, and works with them through challenges, hurdles and objectives.

This year, Endure decided to invest in a new technology used in Apple factories. This technology processes apples, and outputs some data in numbers. One of the very first problems they faced, is bucketing apples into 3 buckets based on their size.

Because everyone's taste is different, some people prefer small apples, others prefer medium ones and the rest prefer large apples. Your task is to contribute to this technology by counting the number of small, medium and large apples produced by the apple factory.

You will be given N apples, each with a size S_i. Apples are considered small if their size is less than or equal to X. They are considered medium if their size is less than or equal to Y and they're not small. Otherwise, they are considered large apples.

In other words:

- Small apple sizes $\subset [1, X]$
- Medium apple sizes $\subset]X, Y]$
- Large apple sizes $\subset]Y, \infty[$

Given N, their sizes, X and Y, calculate A, B, C; the number of small, medium and large apples, respectively.

Input

The first line of input has a single integer N, $1 \leq N \leq 10^5$ – the number of apples.

The second line has 2 space-separated integers X and Y, $1 \leq X < Y \leq 10^5$.

The third line has N space-separated integers, each representing the size S_i of the i_{th} apple. $1 \leq S_i \leq 10^5$.

Output

Output 3 integers; A, B and C, each representing the number of small, medium and large apples, respectively.

Examples

apples.in	standard output
4 17 21 2 3 25 24	2 0 2
5 80 98 100 99 100 69 100	1 0 4

Problem F. Fouad Wears Suits

Input file:	`fouad.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`Black`

Eng. Mohamed Fouad, the ACPC Executive Director, recently got invitations for 3 weddings from couples who met at the ACPC: Jemy and Zizi, Ali and Sara, and Osama and Marwa.

For every wedding, Eng. Fouad **must** wear a new suit. At one of the weddings, he tried to count how many suits he will need if all his acquaintances got married. Eng. Fouad does not care much about the Jacket of the suit, as he has many of them. He cares about the number of trousers and shirts, because every combination of them, makes a new suit.

Suppose that Eng. Fouad has X trousers and Y shirts, where every trouser can be matched with at most one shirt (any shirt) and every shirt can be matched with at most one trouser (any trouser). Also, if a shirt is matched with a trouser, that shirt cannot be used with any other trouser at a later wedding.

Eng. Fouad wants to count the maximum number of suits he can come up with so that he can attend the maximum number of weddings without buying new shirts and trousers.

Given the number of trousers and shirts: X and Y, output the number of suits Eng. Fouad can make out of them.

Input

The input is one line that has 2 space-separated integers; X and Y – the number of trousers and shirts, respectively, where $1 \leq X, Y \leq 1\,000$.

Output

Output a single integer, the number of suits that Eng. Fouad can make to attend the maximum number of weddings.

Examples

fouad.in	standard output
7 3	3
55 89	55
500 500	500

Problem G. Game of Science

Input file:	`game.in`
Output file:	`standard output`
Time limit:	`1.5 Seconds`
Balloon Color:	`Blue`

The ACPC scientific committee members were wondering what kind of games they can play, where the only elements present in the game, are numbers. So, they came up with the following game.

There are 2 players playing a game and alternating moves. Initially, they have a number X that they put on the table. That number can be splitted later to multiple numbers as players make their moves in the game.

In their move, each player can select a number from the table. Let's call this number N. If the number N contains any 0 digits, the player can (if they want to) split it before making their move.

Splitting can happen by removing the digit 0 from the number, and then obtaining 2 numbers, the left and right parts of the original number (with respect to the digit 0) to put them on the table. Note that if the digit 0 is the rightmost digit, only the left part remains (e.g. when $N = 10$).

Also, if the number contains several 0 digits, the player in turn can split on any number of zeros they want. This means they can choose to not split at all, or split on one of the zeros (and have 2 new numbers on the left and right), or split on any 2 zeros (and have 3 numbers on the right, left and in the middle), and so on.

After splitting (if any), the player converts the number into several numbers, removes the original number from the table, and puts the new numbers. To finish their turn, they must minus exactly 1, 2, or 3 from exactly one of those numbers on the table.

The player who cannot make a move, loses the game.

Given X, and assuming optimal play by both players, we want to know who is winning? Is it the first player to win? or the second?

Input

First line will be $1 \leq T \leq 10^6$, the number of test cases.

T lines follow: Each line has one integer $1 \leq X \leq 10^6$

Output

For each test case, output one line, containing "First" if the first player wins the game, or "Second" if the second player does.

Example

game.in	standard output
3 1 4 30204	First Second First

Note

For example, if X = 1020304;

Then the player in turn can make it 1020303, 1020302, 1020301, without splitting.

Or 1, 20304 and then choose a number to minus from, e. g. take the number 20304 and converts it to 20303 or 20302 or 20301.

Or 102, 304 and then choose a number to minus from.

Or 10203, 4 and then choose a number to minus from.

Or 1, 2, 304 and then choose a number to minus from.

Or 1, 203, 4 and then choose a number to minus from.

Or 102, 3, 4 and then choose a number to minus from.

Or 1, 2, 3, 4 and then choose a number to minus from.

Problem H. Holidays for the Volunteers

Input file:	`holidays.in`
Output file:	`standard output`
Time limit:	`1.5 Seconds`
Balloon Color:	`Pink`

The ACPC volunteers have been working a lot to make the ACPC happen. They have not taken a single vacation in a long time. As the ACPC time approached, they started daydreaming about their vacations. One of the dreams was the following:

You work for a company in the outer space which gives you a salary of A dollars per day, and if you work for K days in a row, you will get a bonus of B dollars.

As years in the outer space are not the same on Earth, the year at this company consists of N days (numbered from 1 to N). This year, there are M holidays, which initially you can't work on.

However, you can choose at most Q holidays and work on them.

You want to buy a fancy car that costs S dollars, so you want to choose Q holidays in an optimal way to earn S or more dollars as soon as possible.

In other words, you want to know the smallest index of the day on which you would have earned S or more dollars. If that day does not exist, print -1.

Note that different blocks of K days do not overlap, which means that if you work for $2 \times K$ days, you will get $2 \times B$ dollars and not $(K + 1) \times B$ dollars as bonus.

Input

You will be given T test cases.

Each test case has:

- The first line has 7 space-separated integers:
  - N – the number of days in a year $1 \leq N \leq 10^9$.
  - M – number of holidays, $1 \leq M \leq 5\,000$.
  - K – number of consecutive days in a row. $1 \leq K \leq 10^8$.
  - Q – number of holidays you can choose to work on. $1 \leq Q \leq M \leq 5\,000$
  - A – your salary per day. $1 \leq A \leq 10^9$.
  - B – the bonus, $1 \leq B \leq 10^9$.
  - S – the price of the fancy car, $1 \leq S \leq 10^{18}$.
- The second line has M space separated integers $H_1, H_2, H_3 ... H_M$, where H_i is the index of the i^{th} holiday in the year. It's guaranteed that these indices are distinct. $(1 \leq h_i \leq N)$.

It is guaranteed that the sum of each of N, M, K, Q, A, B, S over all test cases will not exceed its maximum value.

Output

Output the smallest index of the day at which you can earn S or more dollars.

Print -1 if you can't earn S dollars in any way.

Example

holidays.in	standard output
1 10 2 3 1 1 2 6 2 4	5

Problem I. Index of Fame

Input file:	`fame.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`Cyan`

The ACPC Media team members, decided to make videos and publish them online. So, from now on, we will call them video creators.

Each of our video creators has a fame index which indicates their number of fans (an integer ≥ 0) they have. Then, they decided to make a new channel that gathers all of them, so they put themselves into one or more subgroup(s), with some restrictions:

- A video creator will join only 1 subgroup.
- Each sub-group **must** have unique fame indices. i.e. No 2 video creators with the same fame index are allowed to be in the same sub-group.
- Each sub-group has at least two video creators.

Then, they started making videos using **each possible combination** of the video creators **only once**, respecting one condition: **in each video, one and only one video creator from each subgroup will participate**. If there is only one subgroup, then only one video creator will be there in each video.

After making the videos, each video got a number of views that is equal to 2 to the power of the sum of fame indices of the video creators who participated in it.

We don't know the original number of video creators, their fame indices nor the number of subgroups. You are only given the sum of all views in this channel V, and you are asked to print how many possible scenarios are there. If there's no solution print 0.

Two scenarios are different if:

- The number of subgroups is different.
- The subgroups themselves are different.
- The order of subgroups is different (see the note below).

Two sub-groups are different if:

- The fame indices of the video creators in the subgroups is different.
- The number of video creators in the subgroups is different.

Note that the order of video creators' fame indices in a subgroup does not matter.

Input

The first line of input has one integer T ($1 \leq T \leq 100$) – the number of testcases.

In each of the following T lines, one integer is given for each test case V, where $0 \leq V \leq 10^9$.

Output

For each test case, output one line, the number of possible scenarios.

Example

fame.in	standard output
9 0 1 2 3 4 5 6 30 225	0 0 0 1 0 1 1 5 26

Note

For example: for $V = 30$:

There are 5 possible scenarios:

1. Subgroup #1: {1, 2, 3, 4}
2. Subgroup #1: {0, 2}, Subgroup #2: {1, 2}
3. Subgroup #1 {1, 2}, Subgroup #2: {0, 2}
4. Subgroup #1 {1, 3}, Subgroup #2 {0, 1}
5. Subgroup #1 {0, 1}, Subgroup #2 {1, 3}

{Subgroup #1: {0, 1, 2, 3}, Subgroup #2: {1}} is not a valid scenario, since the second subgroup has only one video creator.

Problem J. Just Count the Strategies

Input file:	`bios.in`
Output file:	`standard output`
Time limit:	`1 Second`
Balloon Color:	`Gold`

The ACPC System team, under the leadership of Combo, designed a perfect System for the ACPC.

Then, they handed the system to you. Before you started trying the system, you wanted to install a new Operating System. So, you created a bootable USB drive and rebooted the PC, but you discovered that you forgot the shortcut key to change your BIOS boot device.

The key is one of the keys: {F1, F2, ..., F12}, and you must press it as the computer boots. If the correct key is pressed, you enter BIOS, otherwise you fail and you need to restart your computer. You are willing to use your toes if needed (in addition to your fingers), so that you can press **at most** K keys at a time.

You want to know which key is the correct one, so you develop a **strategy** that minimizes the number of restarts needed in the worst case until you **determine with certainty** which key is the correct BIOS key. Such strategy depends on the maximum number of keys you can press at the same time K.

A strategy is a set of instructions that you can follow, to determine the correct bios key.

There exist strategies that do not minimize the maximum number of restarts, but we are not interested in them. We are interested in **optimal strategies** (the ones that minimize the maximum number of restarts needed to find the correct key) only.

Given K (the maximum number of keys you can press at the same time), count the number of optimal strategies.

Since the number of optimal strategies can be very large, compute the number modulo $10^9 + 7$.

Keep in mind that:

- Two strategies are different if:
  - Their instructions are different.
  - The order of instructions is different.
- Two Press instructions are different if:
  - They have different number of keys.
  - They have different keys (even if the number of keys is the same).
- Two press instructions containing the same keys are **not** different regardless the order of the keys, because those keys are pressed at the same time (i.e. in the same press instruction).
- Press operations do not use keys that are known to be incorrect (i.e. you never press on a key that you know - from earlier presses - that is incorrect).
- A strategy never repeats the same press operation, and never performs a press operation that will not add any new information (i.e. pressing on all keys).
- A conditional instruction is **always** in the form of "If bios was entered, do something, else do another thing", it **cannot** be in the form of "If bios was not entered do something, else do another thing".

Input

You will be given a single integer K, where $(1 \leq K \leq 12)$ and K is the maximum number of keys you can press at a time.

Output

Output a single integer, the number of unique optimal strategies modulo $10^9 + 7$.

Examples

bios.in	standard output
1	479001600
6	135103783

Note

Example strategy for $K = 7$:

```
Restart and Press on keys F1, F2, F3, F4, F5, F6, F7
If bios was entered:
  Restart and Press on keys F1, F2, F3, F4
  If bios was entered:
    Restart and Press on keys F1, F2
    If bios was entered:
      Restart and Press F1
      If bios was entered:
        F1 is the key
      Else:
        F2 is the key
    Else:
      Restart and Press F3
      If bios was entered:
        F3 is the key
      Else:
        F4 is the key
  Else:
    Restart and Press on keys F5, F6
    If bios was entered:
      Restart and Press F5
      If bios was entered:
        F5 is the key
      Else:
        F6 is the key
    Else:
      F7 is the key
Else:
  Restart and Press on keys F8, F9, F10
  If bios was entered:
    Restart and Press on F8, F9
    If bios was entered:
      Restart and Press F8
      If bios was entered:
        F8 is the key
      Else:
        F9 is the key
    Else:
      F10 is the key
  Else:
    Restart and Press on F11
```

```
    If bios was entered:
      F11 is the key
    Else:
      F12 is the key
```

This strategy for example, minimizes the number of restarts, no matter what the correct key is. It will take at most 4 restarts to **determine** the correct key.

Another strategy that minimizes the number of restarts when $K = 7$:

```
Restart and Press on keys F1, F2, F3, F4, F5, F6
If bios was entered:
  Restart and Press on keys F1, F2, F3
  If bios was entered:
    Restart and Press on keys F1, F2
    If bios was entered
      Restart and Press F1
      If bios was entered:
        F1 is the key
      Else:
        F2 is the key
    Else:
      F3 is the key
  Else:
    Restart and Press F4, F5
    If bios was entered:
      Restart and Press F4
      If bios was entered:
        F4 is the key
      Else:
        F5 is the key
    Else:
      F6 is the key
Else:
  Restart and Press on keys F7 F8, F9
  If bios was entered:
    Restart and Press on F7, F8
    If bios was entered:
      Restart and Press F7
      If bios was entered:
        F7 is the key
      Else:
        F8 is the key
    Else:
      F9 is the key
  Else:
    Restart and Press on F10, F11
    If bios was entered:
      Restart and Press F10
      If bios was entered:
        F10 is the key
      Else:
        F11 is the key
    Else:
      F12 is the key
```

Problem K. Kontestants and Group Photos

Input file:	`polygon.in`
Output file:	`standard output`
Time limit:	`4 Seconds`
Balloon Color:	`Yellow`

The ACPC group photo is a tradition that the media team takes seriously every year, and they brainstorm for the best photo idea. This year, the idea is to form every possible convex polygon, whose sides are the contestants themselves. A convex polygon is a simple polygon (non-self intersecting) in which all interior angles are less than 180°.

The contestants are arranged into N groups, where the i^{th} group has A_i contestants (We assume that each contestant takes exactly 1 length unit). A group of contestants **must all** stand next to each other in a straight line and cannot bend in any way. Also, no 2 groups can overlap. How many unique photos can the media team capture?

In other words, What is the number of different subsets of the given lengths that can be used to form a convex polygon? As the number of subsets can be very large, we want it modulo $10^9 + 7$.

Note that: Two polygons are considered the same if the indices of groups used to form one polygon is a permutation of the indices of groups used in the other polygon.

Input

The first line has a single integer N, $1 \leq N \leq 5 \times 10^3$ – the number of contestant groups.

The second line has N space-separated integers, where the i^{th} integer is the length of the i^{th} group A_i, $1 \leq A_i \leq 5 \times 10^3$.

Output

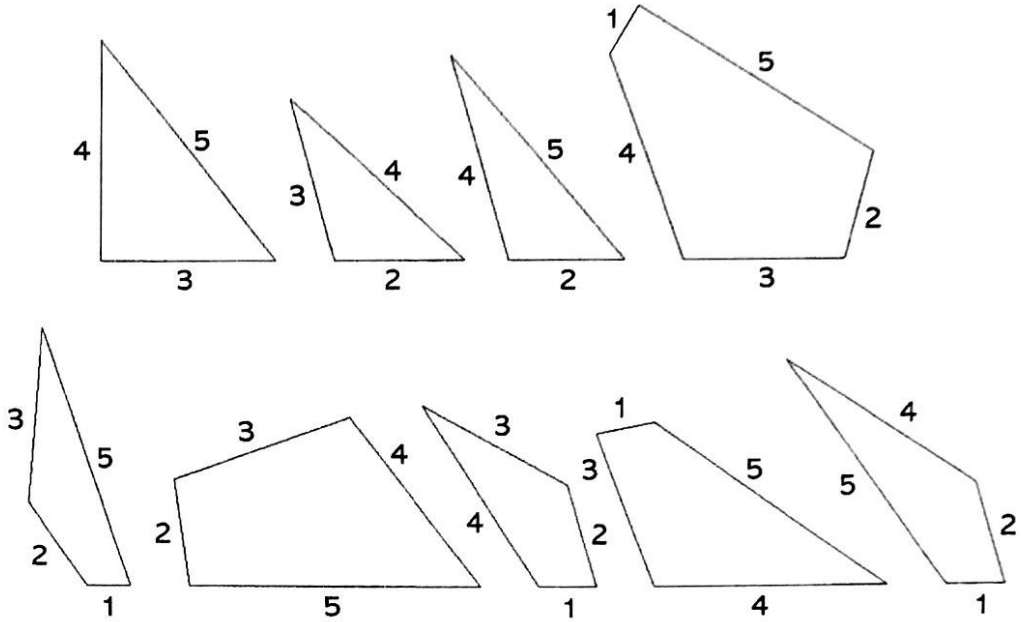
The output is a single integer P, which is the number of unique convex polygons that can be formed by the given groups modulo $10^9 + 7$.

Example

polygon.in	standard output
5 1 2 3 4 5	9

Note

An illustration of the sample test case (all the possible convex polygons) is shown in the figure below:


4
5
3
3
4
2
4
5
2
1
5
4
2
3
3
5
2
1
3
4
2
5
3
4
2
1
1
5
3
4
2
5
4
2
1

Problem L. Looking for the Route to ICPC

Input file:	`cars.in`
Output file:	`standard output`
Time limit:	`8 Seconds`
Balloon Color:	`Rose`

Some contestants have just qualified to represent the region in the ICPC. While this is a great achievement, and a greater responsibility, these contestants have to reach the ICPC venue driving cars.

The world is modelled as a weighted tree of N nodes -numbered from 1 to N- and rooted at node 1. The ICPC venue is at the root. Every time a qualified contestant starts driving, they try to reach the highest node they can (closer to the ICPC venue), park it there and continue the rest walking. A contestant can only move from a node upwards, and the edge connecting the node to its parent has to have a weight W that is smaller than or equal to the current F liters of fuel they currently have in their car.

In other words, the condition $W \leq F$ must hold for every edge with weight W that they cross. Crossing this edge consumes W liters from the cars' tank, but gives the contestant $2 \times W$ liters as a reward.

If a contestant reaches a node where another car has been previously parked (because it could not go higher), they **must** use whatever number of fuel liters is left in it. If a contestant cannot go any further up (because they have reached the ICPC venue or because that the next edge is greater than the amount of fuel they have in their car), then that contestant parks the car at that node, where it will affect the next contestants driving through this node.

You are currently presented with Q queries of two types:

- $1\ V\ W$ – Update the weight of the edge connecting the node V to its parent to W.
- $2\ V\ F$ – A contestant will start driving a car from node V towards the ICPC venue with F liters of fuel. Print the node at which they will park their car.

Input

The first line of input contains two space-separated integers: N and Q ($1 \leq N, Q \leq 500\ 000$) – the number of nodes and queries, respectively.

The next $N-1$ lines describe the tree. In the i^{th} line, you are given P_i – the parent of the $(i+1)^{th}$ node and W_i – the weight of the edge connecting this node to its parent. ($istartsat1$).

Then, Q lines follow, where each line describes a query of one of these forms:

- $1\ V\ W$, where $1 \leq V \leq N$, $1 \leq W \leq 10^9$.
- $2\ V\ F$, where $1 \leq V \leq N$, $1 \leq F \leq 10^9$.

Output

Print the the highest node the contestant will reach for each of query of the second type.

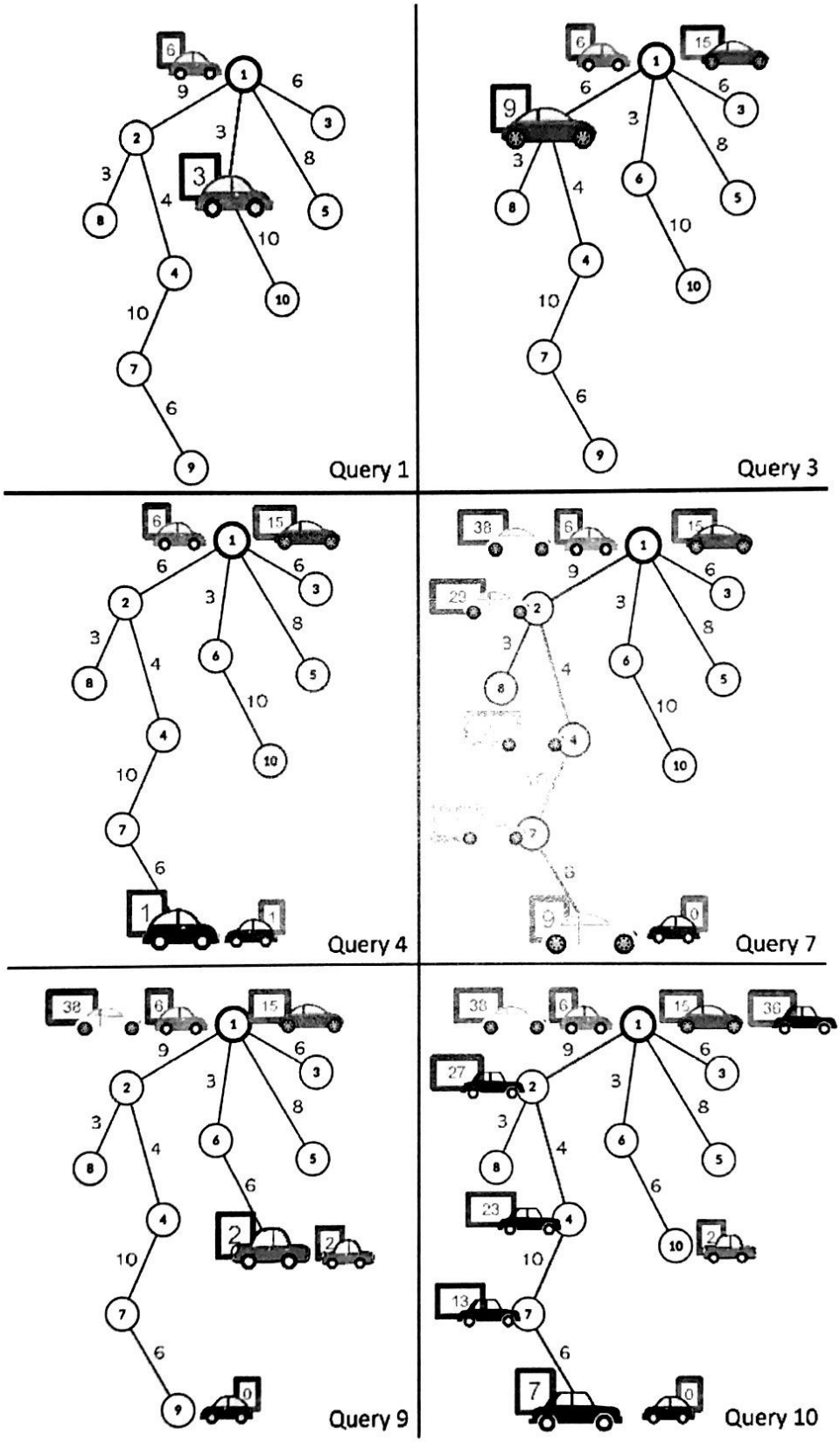
Example

cars.in	standard output
10 10 1 9 1 6 2 4 1 8 1 3 4 10 2 3 7 6 6 10 2 6 3 1 2 6 2 2 9 2 9 1 1 3 6 1 2 9 2 9 8 1 10 6 2 10 2 2 9 7	1 1 9 1 10 1

Note

When a car passes by a node, it will empty all of the parked cars' tanks in its path and take that fuel for itself, even if it doesn't need them at all.

The queries of the second type in the sample test case are visualized below:


Query 1
Query 3
Query 4
Query 7
Query 9
Query 10