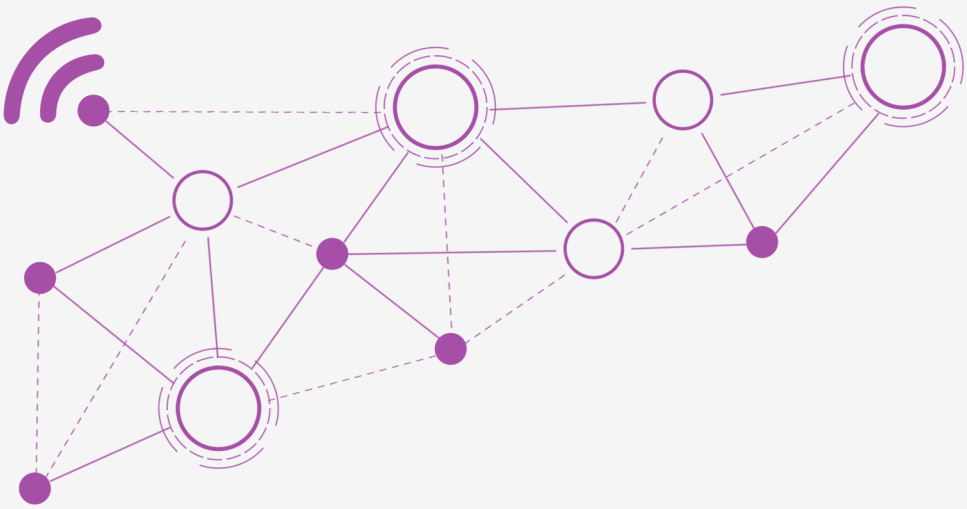
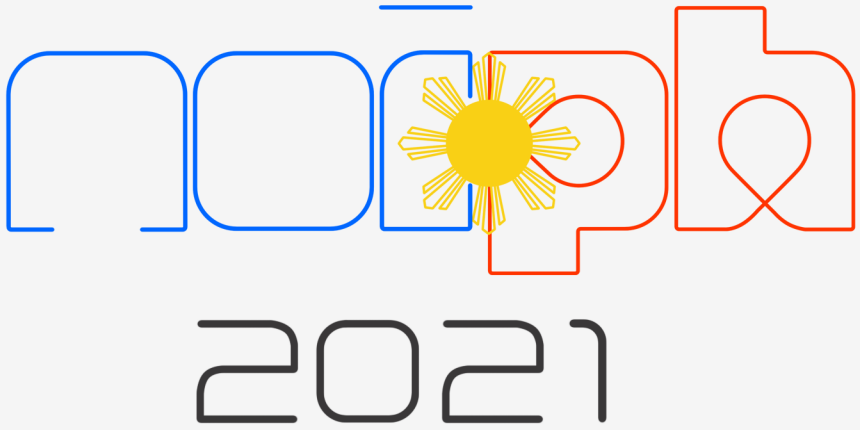




National Olympiad in Informatics

Finals Round 1

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use `BufferedReader` and `PrintWriter`.
  - In C/C++, use `scanf` and `printf`.
- Good luck and enjoy the problems!

Important! Read the following:

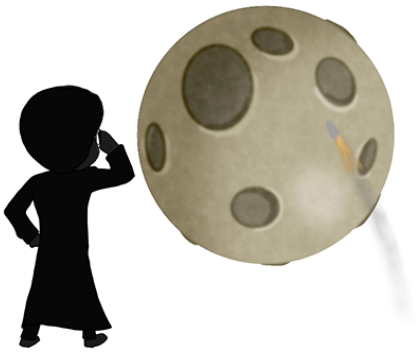
- Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is **strict**. Follow these guidelines strictly:
  - It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "Please enter t".

  Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- For Java, do not add a package line at the top of your code. Otherwise, your submission will receive the verdict "*Execution failed because the return code was nonzero*" or "*Compilation failed*".
- Only include one file when submitting: the source code (.cpp, .java, .py, etc.) and nothing else.
- Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In Java, use `BufferedReader` and `PrintWriter`.
  - In C/C++, use `scanf` and `printf`.

  We recommend learning and using these functions during the Practice Session.
- Good luck and enjoy the contest!

Problem A
To the Moon!

Time Limit: 2 seconds

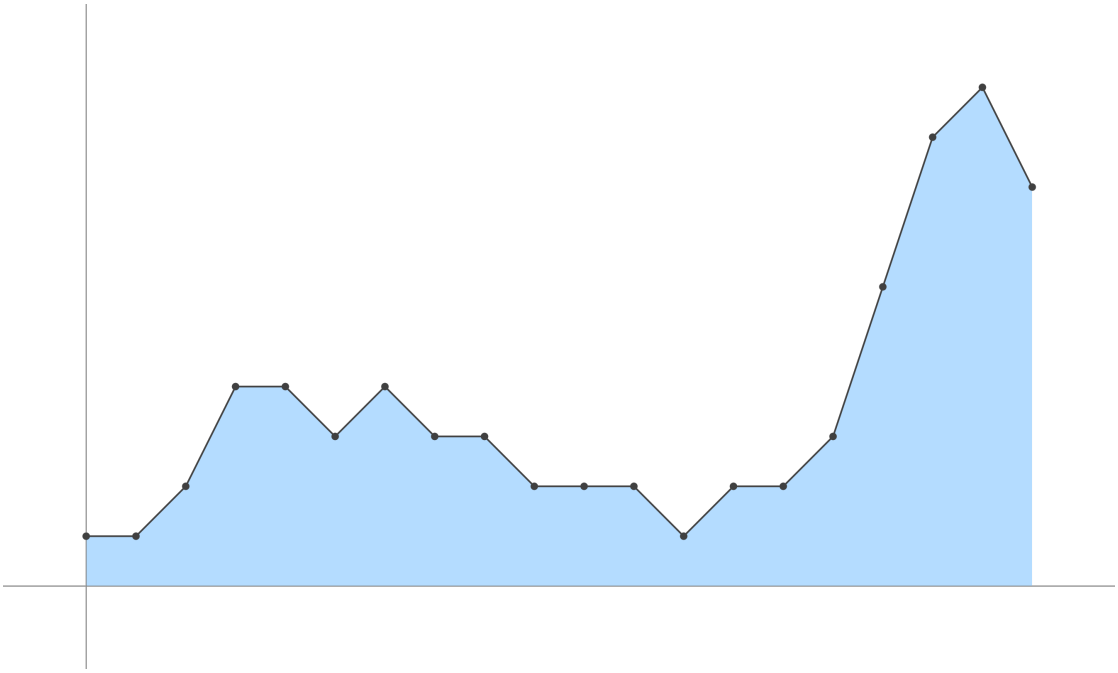


Chinese-Filipino entrepreneur Tess La made a small fortune by selling protective face coverings that comfortably cushion people's noses (what she calls the *Ilong Mask*). Just like any capitalist, her next step is to turn her small fortune into a **big** fortune, and she thinks she will do that by investing in cryptocurrencies!

What are cryptocurrencies? Well, they're called *currency* because they're the hot new trend that everyone is bandwagoning on, and they're called *crypto* because how they work is a total mystery to our monkey brains.

Tess La is also aware that crypto-mining eats up an insane amount of energy, accelerating global warming and the eventual flooding of the Earth. Tess La hopes to get rich quickly enough to afford an escape rocket *to the moon* when the Earth succumbs to a natural apocalypse.

Tess La hopes to manipulate people's opinions on the Pac Token, the Philippines' premier cryptocurrency. She has n positive integers $A_0, A_1, \ldots, A_{n-1}$, corresponding to the price of the Pac Token over time. She will release an infographic of this information shown on a *line chart*. The chart consists of the points (t, A_t), for each $0 \leq t < n$, with line segments connecting each adjacent pair of points.



The *opinion region* is the region between the x-axis and the segments of the line chart, from $x = 0$ to $n - 1$. The opinion region is highlighted blue in the image above. Tess La knows that the general population's sentiment towards Pac Tokens is directly proportional to the *area* of opinion region. She is capable of manipulating the data by *rearranging* the order of the elements of A. Tess La is curious to see just how hard she can potentially push her agenda about Pac Tokens. Find the minimum and maximum possible areas of the opinion region, among all possible permutations of the elements of A.

Input Format

The first line of input contains a single integer n, the number of positive integers Tess La has. Then, it is followed by a line containing n space-separated integers $A_0, A_1, \ldots, A_{n-1}$, the n data points in their original order.

Output Format

Output a line containing two numbers separated by a space—the minimum possible area and the maximum possible area of the opinion region. Your answers will be considered correct if they are within 10^{-3} of the correct answers (in other words, make sure to print at least 3 digits beyond the decimal point).

Constraints and Subtasks

For all subtasks
$2 \leq n \leq 2 \times 10^5$ $1 \leq A_t \leq 10^7$ for all $0 \leq t < n$

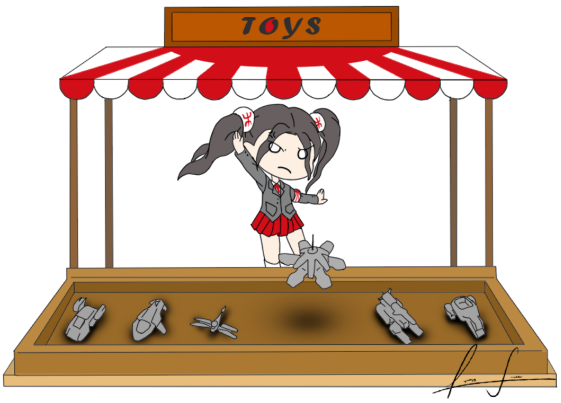
Subtask	Points	Constraints
1	**25**	$2 \leq n \leq 8$
2	**25**	$2 \leq n \leq 16$
3	**50**	$2 \leq n \leq 2 \times 10^5$

Sample I/O

Input	Output
3 1 2 3	3.500 4.500

Problem B
Psari-Psari

Time Limit: 1 second



Aling Wata's sari-sari store has always been well-known around her community. However, when her daughter Dominique Wata took over, it suddenly started attracting *international attention*. People from all over the globe would visit their store, just to buy some Sprite and a small Chippy Red.

But why? What could cause such a change? The products are the same as they've always been, and they've not gone down in price or up in quality. No, the change was not in the wares, but in the *tindera* herself. You see... D. Wata is psychic! This transforms their humble sari-sari store into a paranormal *psari-psari store*.

D. Wata has this strange ability. Based only on how much money a customer gives her, she is *usually* able to instantly know how much change they are to receive—without even knowing what item they are buying! For example, a customer handed her four 20-peso bills, and she instantly knew to give them 10 pesos change—without even seeeing that the item they bought was indeed worth 70 pesos!

However, it turns out that Dominique Wata doesn't truly have magic powers... even better, she has **mathemagic** powers. Dominique only needed to make the following crucial observation: When paying, customers usually never give more coins and bills that they are required to. More specifically, when a customer pays for a certain amount, it is such that no proper subset of the coins and bills they give can also be used to pay for the amount. For example, if a customer is paying for an item worth 70 pesos, they will not give five 20-peso bills, because four 20-peso bills would have been enough to pay for the item.

As an illustration of how D. Wata's power works, suppose the store only sells two items: one worth 70 pesos and one worth 90 pesos. Then, if a customer gives five 20-peso bills, then D. Wata would know to give them 10 pesos in change. As an

example of when her power *doesn't* work, if a customer gives two 50-peso bills, then D. Wata would need to ask to clarify which item they are buying. Morevoer, if a customer gives six 20-peso bills, then D. Wata would also be confused, because that is more bills than necessary for *either* item in the store.

To make the trick more likely to work, Dominique Wata later enforced a rule that **each customer may only buy one item** per visit to the psari-psari store. Even then, the trick still isn't foolproof. To summarize, the trick works if and only if there exists **exactly one** item in the store that can be bought with the coins and bills given by the customer but *not* by any subset of them.

You are given the prices of the n items in D. Wata's store. You need to answer two types of questions:

- $type = 1$: You are given the money that c customers use to pay for their respective purchases. Using this information, help D. Wata predict how much change to give each customer.
- $type = 2$: There are m distinct denominations of currency. Given the value of each denomination, help D. Wata count the number of *collections* of currency such that the change can be predicted. Two collections are considered the same if and only if each denomination of currency appears the same number of times in both collections. In other words, order doesn't matter. Since the answer may be very large, you need to find this count modulo 998244353.

Input Format

The first line of input contains a single integer, *type*, denoting the question type. Note that *type* will be either 1 or 2.

- If $type = 1$, the next line contains two space-separated integers n and c, the number of items and the number of customers, respectively.

  The following line contains n distinct space-separated integers denoting the prices of the items. Then the description of the c customers follow. The first line contains an integer k denoting the number of bills/coins this customer used to pay. The second line contains k space-separated integers denoting their corresponding values.

- If $type = 2$, the next line contains two space-separated integers n and m, the number of items and the number of denominations, respectively.

  The following line contains n distinct space-separated integers denoting the prices of the items.

  The following line contains m distinct space-separated integers denoting the values of the denominations.

Output Format

- If $type = 1$, print c lines, one for each customer. For each customer, if the change can be predicted, print a single integer denoting the change, otherwise, print a single line containing the string "`ANO YUN`" (without the quotes).
- If $type = 2$, print a line containing an integer denoting the number of nonempty collections of currency there are such that the change can be predicted uniquely, modulo 998244353.

Constraints and Subtasks

Here,

- V denotes an upper bound on the price and currency values appearing in the input. Every price or currency value appearing in the input will be between 1 and V, inclusive.
- K denotes the sum of the ks across all customers (for $type = 1$).

For all subtasks
The prices of items are distinct. For $type = 2$, the currency values are distinct.

Subtask	Points	Constraints
1	**30**	$type = 1$ $1 \le n, k \le 16$ $1 \le c \le 160$ $K \le 160$ $V = 10^8$
2	**20**	$type = 1$ $1 \le n, k \le 120000$ $1 \le c \le 120000$ $K \le 120000$ $V = 10^8$
3	**25**	$type = 2$ $1 \le n, m \le 4$ $V = 20$
4	**15**	$type = 2$ $1 \le n, m \le 80$ $V = 1200$
5	**10**	$type = 2$ $1 \le n, m \le 2000$ $V = 30000$

Sample I/O

Input 1	Output 1
1 2 4 70 90 5 20 20 20 20 20 2 50 50 9 50 20 10 1 1 5 1 1 1 10 50 20 10 1 1 5 1 1 1 1	10 ANO YUN 0 ANO YUN

Input 2	Output 2
2 3 4 6 9 11 10 5 2 1	33

Explanation

The first sample input is of the first type of question, where we are asked to predict the change of each customer. There are 2 items in the store, and 4 different customers to process. The 2 items in the store are worth 70 and 90 pesos. Then, we proceed with the 4 customers.

- For the first customer, D. Wata can predict that the change should be 10 pesos, because the customer bought the item worth 90 pesos. If they bought the item worth 70 pesos, then they wouldn't have given *five* 20-peso bills.
- For the second customer, D. Wata cannot predict the change, because two 50-peso bills is exactly enough for both the 70 and 90-peso items. Since there are two possibilities, she has to say `ANO YUN`.
- For the third customer, D. Wata was paid exactly 90 pesos, so she knows that the change should be 0 (for the 90-peso item).
- For the fourth customer, D. Wata was paid exactly 91 pesos. In all cases, this is more coins and bills than necessary! Both the 70 *and* the 90 peso item could have been paid for by a subset of the coins and bills handed to her, so she is confused. Since there is no "solution", she has to say `ANO YUN`.

The second sample input is of the second type of question, where we are asked to count the number of nonempty collections of currency there are such that the change can be predicted uniquely by D. Wata's trick. There are 3 items in the store, and 4 denominations of currency. The items in the store are worth 6, 9, and 11, pesos, respectively. The currency have values 10, 5, 2, and 1.

We can count a total of 33 collections of currency such that D. Wata's prediction trick works on it. To give some examples...

- The collection of $\{5, 10\}$ is good, because D. Wata can predict that the customer is buying the 11-peso item.
- The collection of $\{5, 5\}$ is bad, because D. Wata cannot tell whether the person is buying the 6 or the 9-peso item.
- The collection of $\{10, 1, 1\}$ is bad, because that is more money than strictly necessary for *any* of the items. If they were buying the 6 or 9-peso item, just the 10 would have sufficied; if the were buying the 11-peso item, just a 10 and a 1 would have sufficed.

Problem C

Pagtitipon ng Hiwaga

Time Limit: 1 second

Seto Kakaiba and Garfield the Cat are the Philippines' top card game players. They have mastered Tong-its, Pusoy, and Pusoy Dos, and even nonlocal games like Blackjack and all the Hold 'Em variants. Now, having gone undefeated (except by each other) for two decades, the two of them have grown bored and seek a new challenge.

"What if we created a machine that tore a portal into the afterlife so that we could duel against the greatest card game players in history?" Seto suggested.

"No, that sounds like too much work," said Garfield the Cat.

"How about we create an academy whose focus is entirely on playing card games?" Seto suggested. "Or how about we try to play card games on motorcycles?"

"...Why don't we just make our own card game instead?" Garfield responded.

And so they did. Their game was called *Pagtitipon ng Hiwaga*, and it was going to be the greatest card game in the world.

You begin the game with n cards and m mana. Mana is a resource that you use to play cards. The ith card in your hand has mana cost C_i, meaning it can only be played if you have at least C_i mana, and playing it consumes C_i of your mana. You can play as many cards as you want, in any order, one at a time, so long as you have sufficient mana for each card you wish to play. Also, each card can only be played once.

One special feature of *Pagtitipon ng Hiwaga* is that players have power-ups which will give them a special ability throughout the game. One such power-up, which Kakaiba and Garfield are currently playtesting, is called the *Bato ng Salamangkero*. Whenever you play a card from your hand with an **odd** mana cost, all other cards in your hand have their cost reduced by 1, *except* the cards with mana cost already

0. Keep in mind this checks if the modified cost of the played card is odd—after all other cost reductions have been applied—and *not* its original cost.

Note that the power-up is always active. You cannot choose to deactivate it. In other words, whenever you play a card with an odd mana cost, *all* other cards will be *forced* to have their mana costs reduced by 1 (except the cards with a cost of 0).

Kakaiba and Garfield are concerned that this card is overpowered because one could potentially end up playing a lot of cards in one turn (possibly even all of them). *Bato ng Salamangkero* could create a strategy that revolves around trying to defeat your opponent in one turn without requiring any cards on the board. Playing cards back-and-forth makes an overall game of *Pagtitipon ng Hiwaga* more fun and compelling, but watching your opponent play 20+ cards in one turn is not particularly fun or interactive.

Let's try to evaluate how good the *Bato ng Salamangkero* actually is in many concrete scenarios. Given n, m, and the costs of the cards in your hand, find the maximum number of cards that you can play.

Input Format

The first line of input contains a single integer t, the number of test cases.

Each test case begins with a line containing two space-separated positive integers n and m, the number of cards in your hand, and the amount of mana that you have available. Following that is a line containing n space-separated integers $C_1, C_2, \ldots, C_n$, the mana costs of the cards in your hand.

Output Format

For each test case, output a single integer, the maximum number of cards that you can play with your given amount of mana, assuming that you have the *Bato ng Salamangkero* power-up.

Constraints and Subtasks

For all subtasks
$1 \le t \le 20$ $1 \le n \le 2 \times 10^5$ $0 \le m \le 10^{15}$ $0 \le C_i \le 10^9$ for all $1 \le i \le n$ Let N be the **sum** of n over all test cases. Then, $1 \le N \le 10^6$

Subtask	Points	Constraints
1	**35**	$1 \le n \le 7$
2	**15**	$1 \le n \le 2000$ $1 \le N \le 2 \times 10^4$ The C_i in each test case are distinct.
3	**15**	$1 \le n \le 2000$ $1 \le N \le 2 \times 10^4$
4	**15**	The C_i in each test case are distinct.
5	**20**	No further constraints.

Sample I/O

Input	Output
1 5 4 1 2 4 5 6	3

Explanation

Play the first card, whose mana cost is 1. We have $4 - 1 = 3$ mana left over. Since 1 is odd, the costs of all remaining cards in your hand are reduced from $2, 4, 5, 6$ to $1, 3, 4, 5$.

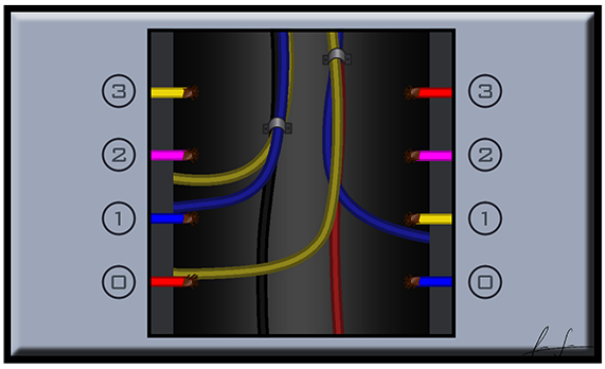
Then, play the second card, whose mana cost was reduced from 2 to 1. We have $3 - 1 = 2$ mana left over. Since 1 is odd, the costs of all remaining cards in your hand are reduced from $3, 4, 5$ to $2, 3, 4$.

Finally, play the third card, whose mana cost was reduced twice, from 4 to 2. We have $2 - 2 = 0$ mana left over. Since 2 is even, no cost reductions happen.

We can show that 3 is the maximum number of cards that can be played in this situation.

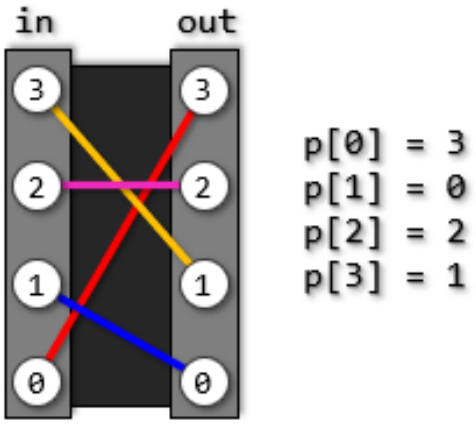
Problem D
Circuit Training

Time Limit: 11 seconds



The Philippines' expedition to the moon on the gossip-powered rocketship is happening soon! The Filipinauts are preparing for the voyage. It is expected that they will have to perform certain *tasks* during the journey to maintain the condition of the ship. One such task that they might need to do is fix the wiring on the ship.

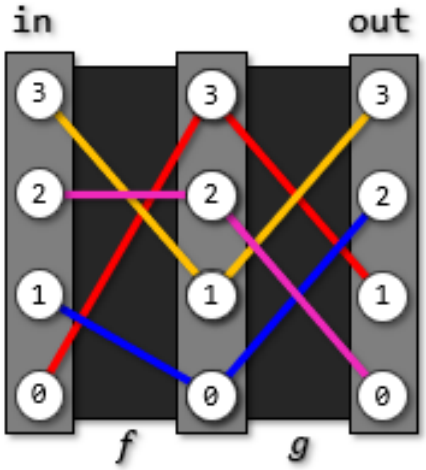
An n-circuit consists of n input nodes, labeled 0 to $n-1$, and n output nodes, labeled 0 to $n-1$. Then, n conductive wires connect each input node to a unique output node and vice versa, sending any charges from the former to the latter. For some circuit p, we use the notation that a charge at input node u is sent to output node $p[u]$. Two n-circuits are equal if for all integers i from 0 to $n-1$, the labels of the output nodes connected to the input nodes labeled i in either circuit are equal.

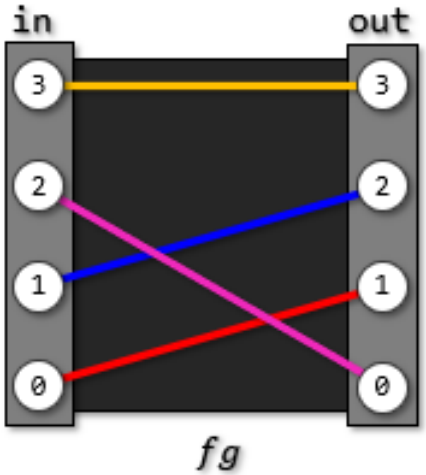


If a circuit breaks, we will need to replace it. We could just carry one of each type of n-circuit with us, but that's ludicrously impractical, even for modestly-sized n.

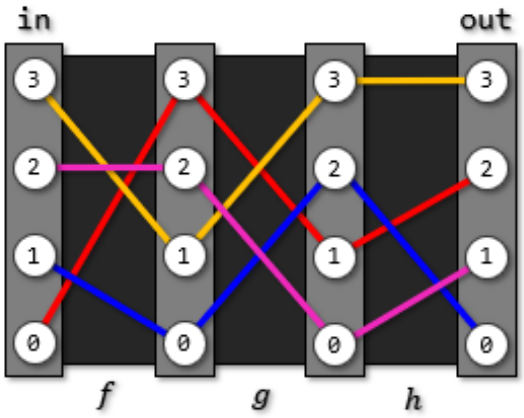
Thus, someone brings up the idea of *chaining* circuits together.

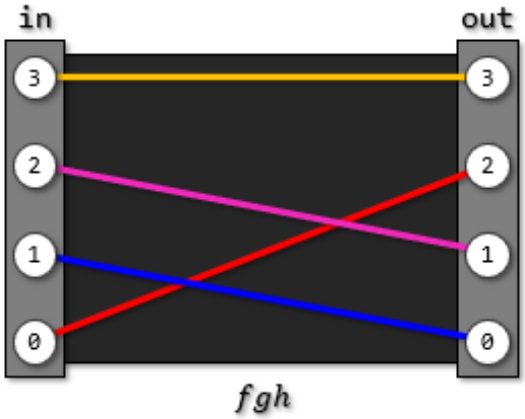
We can chain several n-circuits together to create composite circuits. If we have circuits f and g, the circuit fg is obtained by lining up the output nodes of f with the input nodes of g. This gives us an n-circuit whose input nodes are the input nodes of f, and whose output nodes are the output nodes of g. Charges simply follow the chained-together series of wires. So, a charge at input node u follows the path $u \mapsto f[u] \mapsto g[f[u]]$, and thus is sent to output node $g[f[u]]$. This composite circuit fg can be considered equivalent to a single n-circuit that directly maps $u \mapsto g[f[u]]$.





We can even chain three or more n-circuits together in this manner.

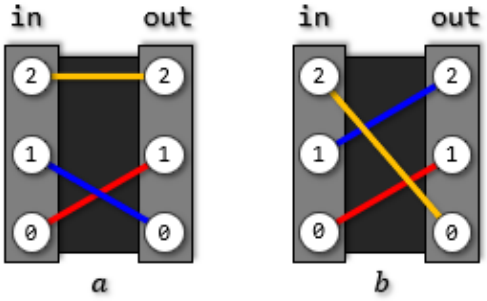




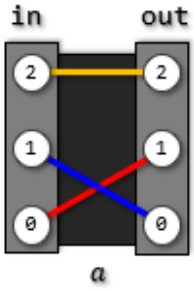
With this technique, we will only need relatively few different types of circuits in order to construct whatever n-circuit we need, so long as we are smart with which few types to bring.

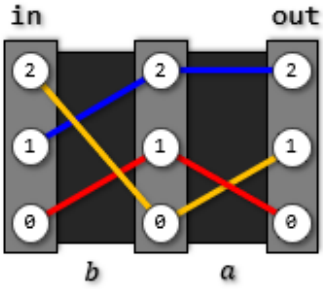
Suppose we had a *toolbox* S that contains q different types of n-circuits, with an unlimited amount of circuits of each type. We call this toolbox *generating* if it is possible to replicate the behavior of *any* n-circuit using only circuits from the toolbox, chained together in some order.

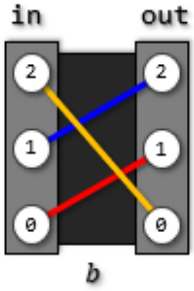
For example, let a and b be the 3-circuits shown below.

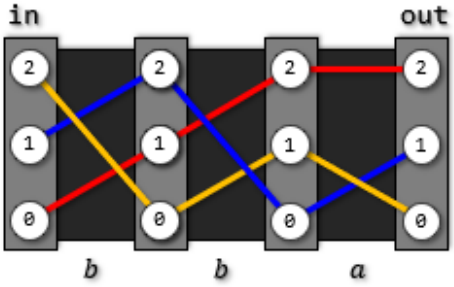


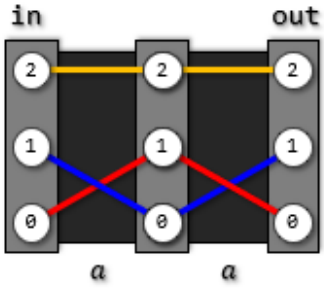
We can show that $S = \{a, b\}$ is a generating toolbox, as the behavior of any 3-circuit can be replicated by composing together combinations of a and b. Note that the constructions shown below do not necessarily use the minimum number of circuits.

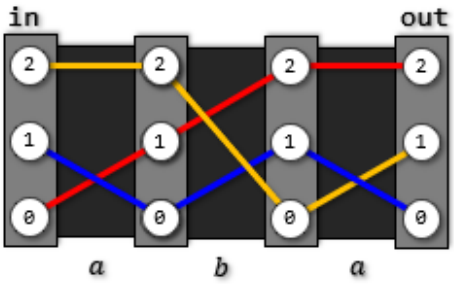












Let's try to build a generating toolbox that contains only some special types of circuits, which R&D has shown to be resilient to damage from spaceflight.

A circuit p is called **quick-returning** if, for every u, **at least one** of $u = p[u]$, $u = p[p[u]]$, or $u = p[p[p[u]]]$ is true.

The *identity circuit* I sends a charge from input node u to output node u, i.e., $u = I[u]$ for all u. A circuit p is called **ultra-quick-returning** if at least one of $p = I$, or $pp = I$, or $ppp = I$ is true.

Note that all ultra-quick-returning circuits are quick-returning, but not vice versa. Note that a and b from the previous example are both ultra-quick-returning.

Your task is to create a *generating toolbox* S using only quick-returning circuits, and using as few quick-returning circuits as possible, and using as many ultra-

quick-returning circuits as possible. In general, the smaller q is, the better, and the more ultra-quick-returning circuits, the better. The exact scoring formula is given below.

You only need to solve this for **a single** n of your choosing, in the range $36 \leq n \leq 50$.

Also, you must convince us that your toolbox really is generating. You will be given several n-circuits, and you must show that each of these can be constructed as some composition of circuits from your toolbox.

Since these constructions can be quite large, you will also be allowed to create some "shortcut" circuits, for convenience. Each shortcut circuit you define must be equal to some combination of members of S and/or other previously defined shortcuts. These shortcuts can then be used in your toolbox constructions. See the Output Format section for more details.

Input Format

The first line of input contains an integer k. The descriptions of k n-circuits follow.

Each n-circuit p is described by a single line containing n, a space, and then n space-separated integers $p[0], p[1], \ldots, p[n-1]$.

Output Format

First, print a line containing an integer n of your choosing, as long as $36 \leq n \leq 50$.

Then print a line containing an integer q ($q > 0$), the size of S. Then print q lines, each defining a new member of S. Each member of S is identified by a single uppercase or lowercase English letter. To define a member p of S, print this identifying letter, an equals sign (=), and a sequence of n space-separated integers $p[0], p[1], \ldots, p[n-1]$.

Then print a line containing an integer s ($s \geq 0$), the number of shortcuts you want. Then print s lines, each defining a new shortcut. Each shortcut is identified by a single uppercase or lowercase English letter. To define a shortcut, print this identifying letter, an equals sign (=), and a string of uppercase or lowercase English letters corresponding to the sequence of n-circuits. Each letter must identify either a member of S or a shortcut that has already been defined.

The rest of the lines will be for showing that your toolbox really is generating.

For every n-circuit p in the input whose n matches the n you printed, print a string of uppercase or lowercase English letters corresponding to a sequence of n-circuits that is equivalent to p. Again, each letter must identify either a member of S or a shortcut that has already been defined.

Ignore the n-circuits that don't match the n you printed.

You may use a dot (‘`.`’) to stand for the empty string, which corresponds to chaining exactly zero n-circuits from your toolbox, which is equivalent to the identity n-circuit. Note that you cannot use ‘`.`’ as part of any other string, e.g., “`ab.ba`” is invalid.

Each uppercase or lowercase English letter can only correspond to at most one definition, either for a member of S or a shortcut.

Each string of uppercase or lowercase letters must have a length of at most 30000.

You may add spaces around each equals sign.

Constraints

- There is only one test file.
- $k = 1200$
- For each n from 36 and 50, there are exactly 80 n-circuits in the input.
- The ns are nondecreasing across all lines of input.

Scoring

Let q be the size of S, and let u be the number of ultra-quick-returning members in S. (Thus, $0 \le u \le q$.) Your score will be dependent on $f(q, u)$, defined as:

$$f(q, u) = \frac{q(q+1)}{2} + q - u.$$

The following is a table of values of $f(q, u)$ for $0 \le u \le q$:

$q \setminus u$	0	1	2	3	4	5	6	$\cdots$
0	0							$\cdots$
1	2	1						$\cdots$
2	5	4	3					$\cdots$
3	9	8	7	6				$\cdots$
4	14	13	12	11	10			$\cdots$
5	20	19	18	17	16	15		$\cdots$
6	27	26	25	24	23	22	21	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

The lower the $f(q, u)$, the higher the score. Here are some more properties of $f(q, u)$:

- If $u_1 > u_2$, then $f(q, u_1) < f(q, u_2)$. In other words, for a fixed q, the more ultra-quick-returning circuits, the better.

- If $q_1 < q_2$, then $f(q_1, u_1) < f(q_2, u_2)$. In other words, a lower q is better, regardless of the number of ultra-quick-returning circuits.

In summary, **your main priority is minimizing q, and your secondary priority is maximizing u.**

Your score is 0 if some n-circuit wasn't constructed successfully, or if your output doesn't follow the output format. Otherwise, your score is computed as follows:

$$\text{score} = \begin{cases} 0 & \text{if } f(q,u) > 1430 \\ 12 & \text{if } 666 < f(q,u) \le 1430 \\ 21 + 69 \cdot \frac{\sqrt{666} - \sqrt{f(q,u)}}{\sqrt{666} - \sqrt{6}} & \text{if } 6 \le f(q,u) \le 666 \\ 95 & \text{if } f(q,u) = 5 \\ 97 & \text{if } f(q,u) = 4 \\ 100 & \text{if } f(q,u) \le 3 \end{cases}$$

Sample I/O

Input	Output
11 6 0 2 4 1 3 5 6 1 3 5 0 2 4 6 2 1 5 0 3 4 6 0 1 2 3 4 5 6 5 4 3 2 1 0 7 2 4 0 3 1 5 6 7 2 4 0 3 1 5 6 7 0 4 1 3 2 5 6 7 4 2 1 3 0 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6	7 4 a = 2 1 4 3 0 5 6 b = 0 2 1 3 4 6 5 A = 0 2 1 3 5 6 4 B = 3 0 2 1 4 6 5 5 Y = babababa s = baababaaa p = baYAbAs y = papaYa o = . popoYo poYopo Yoyo baso . o

Explanation

Note: The sample input doesn't follow the constraints and will not be used to evaluate your submission; it only serves to illustrate the input and output formats.

If we ignore that the input is invalid and we disregard the output requirement that $36 \leq n \leq 50$, then the score is computed as follows. We have $q = 4$ and $u = 2$, so $f(q, u) = 12$, and so the score will be $21 + 69 \cdot \frac{\sqrt{666}-\sqrt{12}}{\sqrt{666}-\sqrt{6}} \approx 87.003$ points.