Problem A. Ancestors And Queries

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	512 mebibytes

You are given a rooted tree with nodes numbered $1, 2, \ldots, n$, where the root is node 1.

You are also given a sequence $a_1, a_2, \ldots, a_m$ of length m, consisting of integers from 1 to n, each element representing a node of the tree.

You have to answer q queries. In each query, you are given an interval of the sequence and a node of the tree. You need to find the maximum node number obtained by taking the LCA of the given tree node and each node selected from the interval of the sequence.

Formally, let $\mathrm{LCA}(u, v)$ denote the Lowest Common Ancestor of nodes u and v in the tree. For a query (ℓ, r, u), you need to compute $\max_{\ell \le k \le r} \mathrm{LCA}(a_k, u)$.

Input

The first line of input contains three integers: n, m, and q ($1 \le n, m, q \le 5 \cdot 10^5$).

Each of the next $n - 1$ lines contains two integers, u and v, representing an edge between nodes u and v ($1 \le u, v \le n$). The resulting graph is a tree.

The next line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$).

Each of the next q lines contains three integers: ℓ, r, and u ($1 \le \ell \le r \le m$, $1 \le u \le n$), representing a query about the interval $[a_\ell, a_r]$ of the sequence and the tree node u.

Output

For each query, output one line containing a single integer: the answer to that query.

Example

standard input	*standard output*
10 12 20 1 10 1 9 9 4 9 5 4 8 4 7 5 2 7 6 2 3 10 8 6 4 3 2 5 7 1 4 6 7 5 8 1 1 12 1 5 6 2 1 3 2 5 5 3 5 7 3 8 12 4 1 5 4 1 1 5 5 6 5 11 12 6 1 2 6 9 12 7 7 9 7 1 4 8 6 11 8 1 1 9 9 10 9 2 12 10 1 1 10	1 1 2 9 3 5 4 9 1 5 7 4 7 9 8 9 1 9 1 10

Problem B. Build The String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	1024 megabytes

Given two words s_1 and s_2 consisting of Latin letters. The first letter in each of the words is uppercase, and the remaining letters are lowercase. It is required to perform compression of these data words, by constructing a string s of minimum length that contains both given words as substrings.

Moreover:

- In the string s, the initial letters from which the substrings corresponding to s_1 and s_2 begin must be uppercase.
- After the initial letter in the substring s corresponding to one of the two given strings, there may be an uppercase letter, which represents the corresponding lowercase letter from the given string. The string s must contain one or two uppercase letters.
- If there are several solutions, output the one that is first among them when ordered alphabetically (keep in mind that any uppercase letter is smaller than any lowercase letter).

A substring of a string s is a string that consists of several consecutive elements of s. In particular, a substring may coincide with the entire string s.

Input

The first line of the input files contains one integer $1 \le t \le 20$ — the number of test cases.

The following are the test cases. Each test case is given on a separate line and consists of the words s_1 and s_2, separated by a space and satisfying the problem condition. The words s_1 and s_2 are non-empty, and the length of each of these words does not exceed 100. The first letter of each word is an uppercase Latin letter, the rest are lowercase Latin letters.

Output

For each test case, output the required string s.

Examples

standard input	standard output
1 Gemini Minibrain	GeMinibrain
2 Chatgpt Cheatgpt Aa Aaa	ChatgptCheatgpt AAa

Problem C. Counting Robot's Paths

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	1024 megabytes

A robot needs to move from point $(0, 0)$ to point (n, n) by executing two types of commands:

- **command X**: the robot moves from point (x, y) to point $(x + 1, y)$
- **command Y**: the robot moves from point (x, y) to point $(x, y + 1)$.

Furthermore, the robot can step on, but cannot pass above the line with the equation $y = x + a$, where a is an integer in the interval $[0, n]$.

Write a program that, given n and a, finds the number of different paths from point $(0, 0)$ to point (n, n) that do not cross above the line $y = x + a$.

Because the output may be too large, print it modulo $M = 10^9 + 7$.

Input

The first line of the input contains two integers n and a ($0 \le n \le 1000$, $0 \le a \le n$).

Output

Print one integer — the answer to the problem.

Example

standard input	standard output
3 1	14

Problem D. DAGPower

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

City K is located between a warm current and mountainous terrain, enjoying unique meteorological conditions that support a thriving traditional handicraft industry. Due to the influence of water vapor, the city's average annual sunshine duration is lower than the national average, so its electricity supply has long relied mainly on thermal power generation. Additionally, its rich historical heritage has made cultural tourism an important pillar industry. Large-scale installation of solar photovoltaic panels might affect the visual appeal of tourist attractions. With the decreasing cost of solar power generation systems, represented by photovoltaic power generation, the city has decided to replace some of its high-consumption, low-efficiency thermal power generation capacity with solar power, in order to better achieve energy conservation, emission reduction, and green low-carbon development goals.

The power transmission network of City K can be viewed as a directed acyclic graph with n nodes and m edges. Nodes without incoming edges represent thermal power plants currently in operation. The remaining nodes with incoming edges represent facilities that require electricity supply. As the operator of City K's power grid, DAGPower Company plans to temporarily convert some thermal power plants into solar power plants for a trial grid operation.

Let S_t denote the set of nodes that remain as thermal power plants during the trial, and S_p denote the set of nodes that DAGPower temporarily converts into solar power plants. DAGPower originally had a temporary conversion plan, but during equipment configuration, they discovered that this plan might easily cause large-scale failures. Specifically, during the trial, there may be facilities that are directly or indirectly connected to both S_t and S_p, receiving power from two different types of power plants simultaneously, which significantly increases the risk of power coupling anomalies and other emergencies. The total risk is estimated as the sum of peak power loads for all such facilities.

However, due to the tight schedule, DAGPower cannot redesign the plan and redeploy equipment from scratch. To minimize the impact on the production and daily life of City K during the trial, DAGPower hopes to develop an emergency conversion plan based on the original scheme. The goal of the emergency plan is to isolate the two power supply types as much as possible, so that the total risk to the city's power grid is minimized. For that, some power plants can be urgently converted to the other type: thermal into solar, or solar back into thermal. However, urgent conversion itself increases the risk, as some power generation capacity would be lost.

Formally, for node i which is a facility, let a_i amperes be its peak power load, and for node i which is a power plant, let a_i amperes be the loss of power generation capacity in case of urgent conversion. The emergency plan works as follows:

- First, DAGPower urgently converts some power plants to the other type.
- After the urgent conversion, let S'_t denote the set of thermal power plants, and S'_p denote the set of solar power plants. For the emergency plan, DAGPower allows S'_t or S'_p to be empty.
- Now, the total risk is estimated as the sum of a_i of facilities that are simultaneously connected to both S'_t and S'_p, plus the sum of a_i for all power plants that were urgently converted to the other type.

Your task is to determine the minimal possible total risk after following an emergency plan.

Input

The first line of input contains two integers, n and m: the number of nodes and the number of direct transmission lines in City K's power network ($3 \le n \le 5000$, $1 \le m \le \min(50\,000, n \cdot (n-1)/2)$.

The second line contains n integers $d_1, \ldots, d_n$, indicating the type of each node:

- If $d_i = -1$, node i is a power-consuming facility.
- If $d_i = 0$, node i is a power plant which will produce thermal power in the original plan ($i \in S_t$).
- If $d_i = 1$, node i is a power plant which will produce solar power in the original plan ($i \in S_p$).

At least one d_i is 1.

The third line contains n integers $a_1, \ldots, a_n$, representing the peak power load (in amperes) of each facility node, or the power generation loss (in amperes) incurred when urgently converting a power plant to the other type ($1 \le a_i \le 40$).

Each of the following m lines contains two integers, u_i and v_i, meaning there is a transmission line in the grid that directly delivers power from u_i to v_i ($1 \le u_i, v_i \le n$). There are no duplicate edges. The input graph is a directed acyclic graph with at least two vertices of in-degree zero. The value d_i equals -1 if and only if vertex i has an in-degree greater than zero.

Output

Print one integer: the total risk (in amperes) of the optimal emergency conversion plan.

Examples

standard input	*standard output*
6 6 0 1 0 1 -1 -1 11 17 1 13 2 28 1 5 2 5 3 5 2 6 3 6 4 6	3
6 7 0 1 0 -1 -1 -1 11 17 1 13 2 28 1 4 2 4 2 5 2 6 3 5 4 6 5 6	12

Problem E. Experimental Astrophysics

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	1024 megabytes

There are n stars located at points with integer coordinates (x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n).

You need to find the lexicographically smallest permutation $p_1, p_2, \ldots, p_n$ such that for every $i \in \{2, 3, \ldots, n\}$ the following inequality holds:

$$\frac{x_{p_{i-1}} + y_{p_{i-1}}}{x_{p_{i-1}} + y_{p_{i-1}} + z_{p_{i-1}}} \leq \frac{x_{p_i} + y_{p_i}}{x_{p_i} + y_{p_i} + z_{p_i}}.$$

A whole group of the scientists worked on this experiment. Can you solve this problem as well?

Input

The first line of the input contains one integer n ($1 \leq n \leq 1000$).

Each of the next n lines contains three integers x_i, y_i, z_i ($1 \leq x_i, y_i, z_i \leq 2 \cdot 10^9$).

Output

Print n integers — the required permutation.

Example

standard input	standard output
3 1 3 1 2 2 1 3 1 1	1 2 3

Problem F. Find The XOR Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Mr. Mej has a strange tree in his garden.

Each vertex of this tree has a label: the vertex with label 1 is the root, and every positive integer appears exactly once as a vertex label.

The vertex with label i has exactly i child vertices, and the labels of these i children are consecutive integers. Formally, let $mn(i)$ be the smallest label among the children of vertex i, and $mx(i)$ be the largest label among them. Then, $mx(i) - mn(i) = i - 1$, and every integer in the range $mn(i)$ to $mx(i)$ appears exactly once.

Moreover, for any $1 \le i < j$, we have $mx(i) < mn(j)$.

These properties uniquely determine the structure of the strange tree. For example, vertex 1 has child 2; vertex 2 has children 3 and 4; vertex 3 has children 5, 6, and 7; and so on.

However, because Mr. Mej does not really like infinite trees, he recently cut almost all vertices from the tree: he only kept the vertices with labels between 1 and n inclusive.

Mr. Mej can draw magical power from this strange tree. Specifically, each vertex on the tree has a *magic value*, which is initially 0. Mr. Mej can choose a vertex x to gather magic, at which time he will obtain the XOR sum of the magic values of all vertices in the subtree of x, including x itself.

Mr. Mej also performs maintenance on the tree. If he chooses a vertex x and a value c, he can cast a maintenance spell that performs a bitwise OR with c on the magic values of all vertices in the subtree of x, including x itself.

Now, Mr. Mej has performed q operations in total, each operation being either maintenance or gathering magic. He wants to know the amount of magic obtained each time he gathers magic.

Input

The first line of the input contains two integers, n and q ($2 \le n \le 10^{18}$, $1 \le q \le 10^6$), representing the size of the tree and the number of operations.

The next q lines each start with an integer op ($op \in \{1, 2\}$), denoting the type of operation:

- If $op = 1$, then two integers x and c follow ($1 \le x \le n$, $1 \le c < 2^{60}$), meaning to apply bitwise OR with c to the magic values of all vertices in the subtree of x.
- If $op = 2$, then one integer x follows ($1 \le x \le n$), meaning to query the XOR sum of the magic values of all vertices in the subtree of x.

Output

For each query (operation with $op = 2$), print a line with a single integer: the XOR sum being asked.

Example

standard input	*standard output*
11 6 1 3 931 1 4 209 1 2 28 2 1 1 8 287 2 4	193 479

Note

In the first example, initially, the magic values of all vertices are 0.

- First operation: apply bitwise OR with 931 to the subtree of vertex 3. The magic values become: 0, 0, 931, 0, 931, 931, 931, 0, 0, 0.
- Second operation: apply bitwise OR with 209 to the subtree of vertex 4. The magic values become: 0, 0, 931, 209, 931, 931, 931, 209, 209, 209.
- Third operation: apply bitwise OR with 28 to the subtree of vertex 2. The magic values become: 0, 28, 959, 221, 959, 959, 959, 221, 221, 221, 221.
- Fourth operation: query the XOR sum of the subtree of vertex 1, which is: $0 \oplus 28 \oplus 959 \oplus 221 \oplus 959 \oplus 959 \oplus 959 \oplus 221 \oplus 221 \oplus 221 \oplus 221 = 193$.
- Fifth operation: apply bitwise OR with 287 to the subtree of vertex 8. The magic values become: 0, 28, 959, 221, 959, 959, 959, 479, 221, 221, 221.
- Sixth operation: query the XOR sum of the subtree of vertex 4, which is: $221 \oplus 479 \oplus 221 \oplus 221 \oplus 221 = 479$.

Problem G. Get Two Strings From One

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 second
Memory limit:	512 mebibytes

Given a binary string of length n, you need to partition it into two subsequences (which may be empty) so that the sum of the two subsequences, each interpreted as a binary number, is minimized. In particular, an empty subsequence is treated as the binary number 0. Output this minimal sum in binary form.

Input

The first line of the input contains an integer t ($1 \le t \le 10^5$), the number of test cases. For each test case:

- The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$).
- The second line contains a binary string of length n.

The sum of all n does not exceed $5 \cdot 10^5$.

Output

For each test case, print a line with the answer in binary form without extra leading zeroes. Print zero as a single '0'.

Example

standard input	*standard output*
3 4 0101 3 000 4 0011	10 0 10

Note

Explanation for the example:

- For the first test case, one optimal partition is this: take the characters at positions 1 and 2 as the first subsequence, and the characters at positions 3 and 4 as the second subsequence. The answer is $01_2 + 01_2 = 10_2$.
- For the second test case, the answer is obviously 0. Note that you must output "0" and not an empty line.
- For the third test case, one optimal partition is this: take the characters at positions 1 and 3 as the first subsequence, and the characters at positions 2 and 4 as the second subsequence. The answer is $01_2 + 01_2 = 10_2$.

Problem H. Hit The Target!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	1024 megabytes

The archer is located at the point with coordinate 0 and shoot in the direction of increasing the Ox axis. The arrow is shot with a speed of p. At the point with coordinate i for $1 \le i \le n$, a target of thickness a_i can be installed (or there may be no target at the corresponding point). When the arrow hits a target, if the speed of the arrow is strictly greater than the thickness of the target, it decreases by a_i, otherwise, the arrow gets stuck in the target.

For each integer i from 1 to n, we define $f(i)$ as the minimum number of targets that must be installed for the arrow to travel a distance not exceeding i (i.e., getting stuck in a target at position i or at a position with a smaller number). Your task is to compute the values of $f(i)$ for all numbers from 1 to n (if for any set of installed targets the arrow flies beyond position i, then $f(i) = -1$).

Input

The first line of input contains two integers n and p ($1 \le n \le 5 \cdot 10^5$, $1 \le p \le 10^9$) — the number of points where targets can potentially be installed, and the initial speed of the arrow, respectively.

The second line contains i integers a_i ($1 \le a_i \le 10^9$). The i-th of these numbers is the thickness of the target that can be installed at point i.

Output

Output n integers $f(i)$, where $f(i)$ is the minimum number of installed targets that stop the arrow at a distance of i or less from the shooting point. If the arrow does not stop at the corresponding distance for any number of targets, output -1.

Example

standard input	standard output
5 11 4 5 2 2 11	-1 -1 3 3 1

Problem I. Incident in the Ruins

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

Yellow sand dances and spreads across the desolate land. The dim red light of a broken signal lamp flickers weakly. Two figures, one tall and one short, walk side by side through a concrete ruin that should not exist. The gray sky enveloping them reflects the feelings in their hearts.

"Are you sure it's around here?" asked O, captain of the special operations team DC-1025.

"Should be. The last location of the distress signal from the two of them appeared near here," said T, looking at the screen of his COMPASS device.

DC-1025 originally planned to investigate this ruin as a whole team, but because O and T received an urgent mission, S and K went ahead alone. Unexpectedly, the two juniors sent a distress signal and then vanished. After handling the urgent mission, O and T immediately rushed to the rescue, but what greeted them was only lifeless broken walls and debris.

"Look at this." Sharp-eyed T noticed a glimmer, bent down, and picked up a small black plastic piece from the sand-covered road. On it was engraved a familiar insignia dotted with dark blue squares — undoubtedly DC-1025's dedicated storage chip "Proof". It seemed this chip was most likely left by S and K; but whether it was intentionally placed or accidentally dropped could not be inferred.

T gently wiped the sand off the chip and inserted it into the matching device he carried. After a few seconds, an unsettling prompt popped up on the screen: chip damaged, read error.

O and T barely managed to read some binary information from the undamaged parts of the nearly-wrecked chip. Below is an example of a piece of binary information read, where "." indicates a binary bit that could not be accurately read:

```
+--------+-------------------------------------------------+
|.00.....|0..10011...........00011....1001.................|
|.00.111.|...10100...10101........01110101...10010...01001|
|.00.011.|...10011...00001...11001.......1................1|
|.00.0...|...................10011...10101...11010...10101|
+--------+-------------------------------------------------+
```

In the information format shown above, the left side is an unsigned integer representing the current timestamp in binary, and the right side is a string recording the specific event.

They read a log recording n events. Each timestamp in the log is an integer represented by m binary bits. Since the event content is difficult to recover directly, they plan to start by restoring the time points of the events. If the event recorder itself had no malfunction, the N events should be recorded in chronological order; a later event's timestamp should be no earlier than a previous one.

Now given these n damaged binary timestamps $s_1, \ldots, s_n$, each of length m, O and T want to know the earliest possible time of the n-th event in the log, and, under the condition that this event occurs at the earliest possible time, the total number of all possible timestamp restoration schemes.

Input

The first line of the input contains two integers n and m: the number of events and the length of each timestamp, respectively ($2 \le n, m \le 200$).

Each of the next n lines contains a string s_i of length m, consisting only of characters '`0`', '`1`', and '`.`', representing the timestamp of the i-th event, where '`.`' indicates a damaged binary bit.

Output

If there exists a way to replace every '.' in all s_i with either '0' or '1' so that the n timestamps form a non-decreasing sequence, then output a string of length m consisting only of '0' and '1' and a non-negative integer, separated by a space. The string should be the smallest possible timestamp that s_n can be restored to among all valid restoration schemes, and the integer should be the total number of valid restoration schemes modulo $10^9 + 7$.

If no restoration scheme exists, output a single integer -1.

Examples

standard input	*standard output*
3 4 ..10 .1.0 ...1	0101 1
3 4 ..1. .1.. ...1	0101 4
3 4 111. 011. 0...	-1
3 8 .00.111. .00.011. .00.0...	00010110 2

Note

In the first example, it can be proved that. among all restoration schemes satisfying the chronological order, s_3 can be restored to 0101 at the earliest. When s_3 corresponds to 0101, there is only one restoration scheme, with timestamps 0010, 0100, and 0101, respectively.

In the second example, s_1 can be either 0010 or 0011, and s_2 can be either 0100 or 0101, so there are 4 valid restoration schemes in total.

Problem J. Just Check The Possibility

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You need to answer several queries.

In each query, given two integers n and m, determine whether it is possible to construct a sequence $a_1, a_2, \ldots, a_n$ consisting of positive integers such that:

- There exists an index j from 1 to n such that $a_j = m$;
- There exists a non-negative integer t satisfying

$$\prod_{i=1}^{n}(a_i + a_{i+1}) = 2^{2t+1},$$

  where $a_{n+1} = a_1$.

If such a sequence can be constructed, output "YES"; otherwise, output "NO".

Input

The first line contains an integer t ($1 \le t \le 10^6$), the number of queries.

Each of the next t lines contains two integers n and m ($1 \le n \le 2 \cdot 10^6$, $1 \le m \le 2^{62} - 1$), meaning you need to check whether there exists a positive integer sequence of length n containing the value m.

Output

For each query, output one line containing the string "YES" or "NO", indicating whether such a construction is possible.

Example

standard input	*standard output*
2 3 3 2 1	YES NO

Problem K. King and Sequence

Input file:	standard input
Output file:	standard output
Time limit:	2 second
Memory limit:	1024 megabytes

King Byteasar obtained a sequence x_i consisting of integers and asks you to determine if there exists an arithmetic progression with an integer common difference for which this sequence is a subsequence. If such a progression exists, find the maximum possible value of the common difference of this progression.

Recall that an arithmetic progression with an integer common difference d and initial term a_0 is an infinite sequence where $a_i = a_{i-1} + d$ for $i > 0$, where d is an integer. A finite subsequence of an infinite sequence is a sequence obtained from a finite prefix of the original sequence by deleting some elements.

Input

The first line of the input contains a single integer n — the number of elements in the sequence x ($2 \le n \le 10^5$).

The second line contains n integers x_i ($-10^9 \le x_i \le 10^9$) — the elements of the sequence x, listed from first to last.

Output

If the sequence x is not a subsequence of any arithmetic progression with an integer common difference, output '?'.

Otherwise, output a single integer — the maximum possible value of d in such a progression.

Examples

standard input	standard output
3 2024 2026 2025	?
2 2 2	0
3 5 4 2	-1
3 2 4 5	1

Problem L. Luxuries and Aesthetics

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Xiao Lan has n gems, where the brightness of the i-th gem is a_i. Now Xiao Lan wants to divide these gems into several groups, so that each gem will belong to exactly one group.

For a group of gems, if the group contains k gems with brightness values $b_1, b_2, \ldots, b_k$, then Xiao Lan defines the *aesthetic value* of this group as $(\sum_{i=1}^{k} b_i)^2/k^2$. For a grouping scheme, Xiao Lan defines its aesthetic value as the sum of the aesthetic values of all groups.

Now Xiao Lan has q questions. Each question asks: if the number of gems in each group must be within $[\ell, r]$, what is the maximum possible aesthetic value among all grouping schemes that satisfy this requirement? Find this value, or determine that there is no such grouping scheme.

Since the answer can be represented as a rational number a/b, for convenience, you only need to output the result modulo $10^9 + 7$. That is, you need to find an integer c in $[0, 10^9 + 7)$ such that $a \equiv b \cdot c \pmod{10^9 + 7}$. Under the constraints of this problem, it can be proved that such a value c always exists and is unique.

Input

The first line of the input contains two integers, n and q ($1 \le n, q \le 5 \cdot 10^5$), the total number of gems and the number of questions.

The second line contains n non-negative integers, where the i-th integer is the brightness a_i of the i-th gem ($0 \le a_i \le 10^8$).

The next q lines each contain two integers, ℓ and r ($1 \le \ell \le r \le n$), meaning the question: when each group must contain between ℓ and r gems (inclusive), what is the maximum aesthetic value achievable among all admissible grouping schemes?

Output

Print q lines. The i-th line should contain an integer, the answer to the i-th question: the maximum aesthetic value (modulo $10^9 + 7$) when each group's size is restricted to the interval given in the i-th question. If no grouping scheme satisfies the condition, output -1.

Example

standard input	*standard output*
8 4 17 4 12 9 15 3 8 6 1 7 5 8 2 4 3 3	864 62500086 500000431 -1

Note

Explanation for the example:

- For the first question, the optimal grouping is to put each gem into a separate group: $\{17\}, \{4\}, \{12\}, \{9\}, \{15\}, \{3\}, \{8\}, \{6\}$. The aesthetic value is 864, which is the maximum.
- For the second question, the only admissible grouping is $\{17, 4, 12, 9, 15, 3, 8, 6\}$. The aesthetic value reaches its maximum $74^2/8^2$, and $(74^2/8^2) \bmod (10^9 + 7) = 62\,500\,086$.

- For the third question, the optimal grouping is $\{17, 15\}, \{12, 9\}, \{8, 6\}, \{4, 3\}$. The aesthetic value reaches its maximum $(32^2 + 21^2 + 14^2 + 7^2)/2^2$, and modulo $10^9 + 7$ the result is $500\,000\,431$.
- For the fourth question, each group must have size exactly 3, but the total number of gems 8 is not a multiple of 3. Hence some gems would always remain ungrouped, so no admissible grouping exists; output -1.