Problem A. Maximize the Ratio

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

Consider the function $R(x)$ of positive integer x as $R(x) = x/S(x)$, where $S(x)$ is the sum of all divisors of x. For example, $R(6) = 6/(1 + 2 + 3 + 6) = 0.5$, $F(7) = 7/(1 + 7) = 0.875$. Given N, find an integer $x \le N$ such that $R(x)$ is maximal. If there is more than one such integer, choose the **maximal** one.

Input

Input consists of one integer N ($1 \le N \le 10^5$).

Output

Print one integer x: the answer to the problem.

Example

standard input	standard output
9	1

Problem B. Premium Train

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

The premium train "Byteland" between Bytesburg and Bytesville is departing from Bytesburg every odd day in the evening. Next day in the morning it arrives to the Bytesville, then in the evening departs back to Bytesburg. So only one train is needed with such a schedule. But sometimes two odd days are going sequentially (for example, Oct 31 and Nov 1). In such cases the railroad company prepares a reserve train for the second odd day.

You are given a day and a month, and you must answer the question: what kind of train will depart from Bytesburg on the next day? Will it be a regular train, a reserve train, or no train at all?

Input

Input contains two integers: the day of month d and the month m ($1 \le d \le 31$, $1 \le m \le 12$). It is guaranteed that the input data is valid: the day in the input exists in the considered year.

Output

Print 0 if no "Byteland" train is scheduled for tomorrow, 1 if it will be a regular train, or -1 if it will be a reserve train.

Examples

standard input	standard output
10 11	1
11 11	0
31 10	-1

Note

In Sample 1 the day is Nov 10; the train departing on Nov 11 is regular. In Sample 2 the day is Nov 11, no train is scheduled for the next day. In Sample 3 the day is Oct 31; a regular train departs on this day, and the next day is the odd day too, so it's time for a reserve train.

Problem C. Interesting Number

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

Bytica calls the integer X *intersting* if the last digit of decimal representation of X is Bytica's favorite digit K, and the number X itself is divisible by K. Given an integer N and a non-zero digit K, find the minimum interesting number greater or equal to N.

Input

Input contains two integers: N ($1 \le N \le 10^9$) and K ($1 \le K \le 9$).

Output

Print one integer: the answer to the problem.

Examples

standard input	standard output
1 1	1
53 3	63

Problem D. Four circles

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

Three circles of radii R_1, R_2 and R_3 respectively are placed on a plane so that each circle touches two others. Let first and second circles touch at point T_1, second and third at T_2, third and first at T_3. Calculate the radius of circumscribed circle of triangle $T_1T_2T_3$.

Input

First line of the input contains three integers R_1, R_2 and R_3 ($1 \le R_1, R_2, R_3 \le 1000$): radii of the circles.

Output

Print the radius of the circumscribed circle of triangle $T_1T_2T_3$ with absolute error 10^{-2} or better.

Example

standard input	standard output
1 1 1	0.577350

Problem E. Build the Graph

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

Consider undirected graphs of N vertices without self-loops, multiple edges and isolated vertices. Your task is to build such a graph which is **not connected** and has the maximum number of edges possible, or tell that there is no such graph.

Input

First line of the input contains one integer N ($2 \le N \le 200$): the number of vertices in the desired graph.

Output

If there is no such graph, print 0. Otherwise on the first line print number of edges M, and on the next M lines print descriptions of edges. Vertices in the graph are enumerated by integers between 1 and N inclusively. Self-loops and multiple edges are not allowed. Each vertex shall be listed in those descriptions at least once. If there is more than one solution, print any one of them.

Examples

standard input	standard output
2	0
4	2 1 2 3 4

Problem F. Fantastic Compression

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Franek had one job: to memorize a permutation P of the sequence $(1, 2, \ldots, n)$. This, however, proved too boring. Instead, he compressed the numbers in a new, fantastic way he devised: he took a small integer k and memorized only the sums of all connected k-length fragments of P. In other words, Franek now has a sequence $S = (S_1, S_2, \ldots, S_{n-k+1})$, where:

- $S_1 = P_1 + P_2 + \ldots + P_k$, - $S_2 = P_2 + P_3 + \ldots + P_{k+1}$, - $\ldots$ - $S_{n-k+1} = P_{n-k+1} + P_{n-k+2} + \ldots + P_n$.

The method swiftly proved not-so-fantastic, though. First, Franek discovered, to his horror, that sometimes there are several permutations which all compress to the same sequence. Also, he is not sure anymore if he remembered the compressed sequence correctly – the initial permutation may now be lost forever!

Given a compressed sequence S, help Franek find all permutations P which correspond to S.

Input

The first line of input contains the number of test cases z ($1 \le z \le 1000$). The test cases follow, each one in the following format:

The first line of a test case contains the length of the permutation n and the small integer k chosen by Franek ($2 \le n \le 25\,000; 2 \le k \le \min(n, 6)$). The second line contains $n - k + 1$ integers: the elements of the compressed sequence S ($1 \le S_i \le 1\,000\,000$).

The total length of permutations in all testcases does not exceed 250 000.

Output

For every test case, output first the number c of permutations that correspond to the given sequence S. In the next c lines, output these permutations in lexicographic order. Every permutation should be given as n integers in a single line, separated by spaces.

Assume that for the given tests, c is never greater than 1 000.

Example

standard input	standard output
2 5 3 8 10 12 5 3 10 10 10	2 1 2 5 3 4 2 1 5 4 3 0

Problem G. Analog Data

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

You are processing data from an analog device. The data consists of N real values. Each value is positive and is represented with no more than 1000 digits and a decimal point. Your task is to print K-th greatest between those values.

Input

First line of the input contains two integers N ($1 \le N \le 1000$) and K ($1 \le K \le N$).

Each of the next N lines contain one value: a positive real number with no more than 1000 digits used, without leading and trailing zeroes.

The decimal point occurs in each value exactly once and may be placed at the beginning (representing an integer part equal to zero), at the end (representing fractional part equal to zero) or between any two digits.

Output

Print the K-th greatest number in the data you received. The number shall be printed in exactly the same format as in the input.

Example

standard input	standard output
5 2 .8 20.19 3.2 22.04 16.	20.19

Problem H. Henry Porter and the Palindromic Radius

Input file:	*standard input*
Output file:	*standard output*
Time limit:	30 seconds
Memory limit:	512 mebibytes

A young wizard, Henry Porter, has just received sad news – the eldest of his family, uncle Markus Radius Palindromus Black, passed away. Uncle Markus had a reputation of being a quite eccentric person, using complicated binary magic, and was also known to be very, very rich.

Black's will states that Henry should inherit his mysterious chamber of treasures. To enter and claim it, however, the young wizard must say the right password H, which is a word of length n, consisting of characters '0' and '1'. Uncle Markus did not tell Henry the password – it certainly wouldn't be his style. Instead, he computed, for every $x = 1, 2, \ldots, n$, the *palindromic radius* p_x – the largest possible integer such that the word $H[x - p_x \,..\, x + p_x]$ of length $2p_x + 1$ centered at $H[x]$ exists and is a palindrome. Henry then only received the values $p_1, \ldots, p_n$. For example, if the password was 10111010, Henry would get the sequence $(0, 1, 0, 3, 0, 1, 1, 0)$.

Henry would prefer Uncle Markus not to test his algorithmic skills while being dead, but, well, there is no one to complain. And he has good friends who can help him! Given the sequence left by Markus in his will, determine all possible passwords that correspond to it. As the will is battered and stained, it might even happen that there is no solution at all.

Input

The first line of input contains the number of test cases z ($1 \le z \le 200\,000$). The test cases follow, each one in the following format:

A test case consists of two lines. The first line contains a single integer n – the length of both the password and Black's sequence ($2 \le n \le 1\,000\,000$). The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le n$) – the palindromic radii for all the characters in the password.

The sum of n values over all test cases does not exceed $5 \cdot 10^7$.

Output

For every test case, output first the number k of possible passwords. If $k > 0$, output in the next k lines all the solutions as

$$0, 1$$

-sequences. The sequences must be given in lexicographic order.

You may assume that k does not exceed 100.

Example

standard input	standard output
1 8 0 1 0 3 0 1 1 0	4 00010000 01000101 10111010 11101111

Problem I. Assimilation

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

An enlightened race of aliens plans to assimilate a star system, to help its inhabitants achieve perfection. They may resist, but – as you are all well aware – resistance is futile.

There are n planets in the system, inhabited by $a_1, a_2, \ldots, a_n$ people, respectively. Aliens start with k assimilation ships and are allowed to make any of the following moves:

- An *invasion* requires landing on a planet with some part of the fleet. The number of landing ships s must be greater or equal to the population m of the planet. After the invasion, these ships disappear, the planet is *conquered* and now has $m + s$ inhabitants.
- A *mobilization* creates, from a conquered planet, a number of new ships equal to the population of the planet. Every planet can be mobilized at most once.

For Aliens, invasions are easy and natural, but mobilizations turn out to be a bit tricky. Help them conquer all the planets in the system with minimal possible number of mobilizations.

Input

The first line of input contains the number of test cases z ($1 \le z \le 30$). The test cases follow, each one in the following format:

The first line of every test case contains two integers n and k ($1 \le n \le 200\,000$; $1 \le k \le 10^9$) – the number of planets, and the size of Aliens' initial fleet. The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$) – the populations of the respective planets.

The sum of n values over all test cases does not exceed 500 000.

Output

For every test case, output a single integer: the minimal number of mobilizations required to conquer all the planets. If such conquest is impossible, output -1 instead.

Example

standard input	standard output
4 3 15 6 5 26 3 15 6 5 27 2 1000000000 500123123 497000000 7 2 6 2 4 1 9 3 12	2 -1 0 4

Problem J. Polygon

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

You are given n segments of lengths $\ell_1, \ell_2, \ldots, \ell_n$, respectively. Determine the largest possible circumference of a convex polygon that can be constructed using these segments (in any order, and not neccessarily all of them). The polygon must be non-degenerate – in other words, its area must be positive.

Input

The first line of input contains the number of test cases z ($1 \le z \le 100\,000$). The test cases follow, each one in the following format:

The first line of a test case contains the number of segments n ($1 \le n \le 100\,000$). In the second line, there are n integers $\ell_1, \ldots, \ell_n$ ($1 \le \ell_i \le 10^9$) – the lengths of the segments.

The sum of n values over all test cases does not exceed $1\,000\,000$.

Output

For each test case, output a single integer – the largest possible circumference of a convex polygon made of given segments. If no such polygon can be constructed at all, output 0.

Example

standard input	standard output
4 6 1 2 3 4 5 6 3 9 5 14 4 5 15 4 6 2 10 11	21 0 15 0

Problem K. Frogs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You may think that frogs are only good for leaping and croaking, but it turns out that they are also quite proficient coders! Your task is to choose three frogs which would form the best team for OpenFrogCup.

In the frogs' favourite pond there are n stones in a row, spaced 1 meter apart from each other. On every stone, a frog sits. Stones (and frogs) are numbered $1, 2, \ldots, n$ from the leftmost to the rightmost one. The i-th frog sits on i-th stone and is described by two parameters: its leap range r_i and its programming skill s_i. The frog can reach any stone which is not farther than r_i meters (in other words, any stone with index j in $[i - r_i, i + r_i]$). Each frog is willing to jump at most once.

The team for OpenFrogCup must consist of exactly three members which can train together. This means that there must be a stone that all three frogs can jump to (allowing zero-length jumps). Determine the largest possible sum of programming skills of such a team.

The limits for the problem guarantee that there always exists at least one possible three-frog team.

Input

The first line of input contains the number of test cases z ($1 \le z \le 30$). The test cases follow, each one in the following format:

The first line of a test case contains an integer n ($3 \le n \le 200\,000$) – the number of stones (and also the frogs). Each of the following n lines contain two integers r_i, s_i ($1 \le r_i, s_i \le 200\,000$) – the range and the skill of the i-th frog, respectively.

The sum of n values over all test cases does not exceed 500 000.

Output

For every test case, output a single integer – the largest possible sum of skills of a three-frog team.

Example

standard input	standard output
3 4 1 39 2 17 4 5 1 40 3 1 10 1 20 1 30 7 5 4 4 3 3 2 2 1 3 2 4 3 5 4	62 60 11

Problem L. Cheese Game

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

After taking part in the annual *Two-player Games and Applied Cryptography Symposium*, Alice and Bob want to relax by playing their favourite game. They have arranged n cheese slices in a row, numbered from 1 do n. As we all know, though cheese is tasty in general, some slices can be better than others – this is why the i-th slice is described by its deliciousness o_i.

Alice starts the game and the players alternate their moves. In a move, a player may eat any set of cheese slices that are still left on the board, providing that the set contains no two neighbouring slices (i.e. numbered i and $i+1$ for any $1 \le i \le n-1$). We assume that the numbers of the slices do not change, so during the game no new neighbouring pairs appear.

Of course, both players aim to maximize the total delicioussness of their eaten pieces. Assuming that they both play optimally, what is the maximal score that Alice can achive?

Input

The first line of input contains the number of test cases z ($1 \le z \le 20$). The test cases follow, each one in the following format:

The first line of a test case contains the number of cheese slices n ($1 \le n \le 100\,000$). The second line contains n integers $o_1, o_2, \ldots, o_n$ ($1 \le o_i \le 1\,000\,000$) – the values of the pieces' deliciousness.

Output

For every test case, output a single integer – the total delicioussness of the slices eaten by Alice, assuming that both players play optimally.

Example

standard input	standard output
2 3 10 10 10 4 1 2 3 4	20 7