Problem A
Dr Who's Banquet

Input File: A.IN
Output File: standard output
Program Source File: A.C, A.CPP, A.JAVA

Dr. Who is organising a banquet, and will be inviting several guests. A guest is happy, if he can chat with a fixed number of other guests. We assume that guests cannot talk to themselves. Help Dr. Who make all his guests happy, if possible, by organising chats between guests.

The program input is from a text file. The file contains several data sets, and each data set is encoded on a line from the file. A data set consists of $n \le 10000$ positive integers $a_1, a_2, \ldots a_n$, separated by single whitespaces. The last integer a_n is immediately followed by the newline character. Each number a_i, with $1 \le i \le n$ is the number of chat partners guest `i` would like to have. We assume that $a_i \le 1000$ for all $1 \le i \le n$. The last data set is followed by the end of file.

If all guests can be made happy, the program output consists of a $n \times n$ matrix `m`, where `m[i][j]=m[j][i]=1` if guests `i` and `j` chat, and `m[i][j]=m[j][i]=0`, otherwise. The matrix will be represented at standard output, as follows: each value `m[i][j]` from a row will be followed by one whitespace (including the last value from the row). Each row will be separated by the newline character. If it is not possible for all guests to be happy, then the program output is the message "`fail`". The matrix and the message are allways followed by an empty line.

In the following table, an input file with 4 data sets and the associated output, is shown.

Sample input	Sample output
3 3 1 1 **4 4 3 3 2 2 2** **3 3 1 1** **2 2 2 2**	**fail** **0 1 1 0 1 0 1** **1 0 0 1 1 0 1** **1 0 0 1 0 1 0** **0 1 1 0 0 1 0** **1 1 0 0 0 0 0** **0 0 1 1 0 0 0** **1 1 0 0 0 0 0** **fail** 0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0

Southeastern European Regional Programming Contest
Bucharest, Romania - Vinnitsa, Ukraine
October 13, 2012

Problem B
Bitris

Input File: B.IN
Output File: standard output
Program Source File: B.C, B.CPP, B.JAVA

There is a set of 2xN cubes. Each cube has an integer ranging from 1 to N assigned to it, labeling each of the cube's sides. Each number is written on exactly two cubes. Cubes are placed one on top of another, in a pile. If two cubes with the same number stand next to each other, they annihilate: both cubes disappear, and cubes standing above come down to fill the space.
Your task is to disassemble the pile – eliminate all cubes. You are allowed to swap any two neighboring cubes. A swap could be done only after all possible annihilations are done.

For example, if N=4 and cubes are standing as you see on the right then you need to make one swap. Cubes with label 3 annihilate immediately; you swap the fourth bottom cube (with label 1) and the fifth bottom cube (with label 4); afterwards, cubes with labels 4 annihilate, followed by cubes with labels 1 and labels 2. Other option is to swap third and fourth bottom cubes (in this case cubes with labels 1 and 4 annihilate at same time, followed by cubes with label 2), or second and third.

2
4
1
3
3
4
1
2

If N=3, and cubes are standing like shown on the right, you need to perform 3 swaps. One way to solve is to swap fifth and sixth cubes, then fourth and fifth; cubes with labels 2 annihilate; then swap second and third; other cubes annihilate simultaneously.
The task is to find the minimal number of swaps, such that all cubes are eliminated.

2
3
1
2
3
1

Input data. The first line in the input file contains the positive integer N, 2<=N<=100000. The second line contains the labels of all cubes, bottom up, split by spaces.
Output data. The only line in the output should contain one non-negative integer M – the minimal number of swaps necessary to destroy all cubes.

Sample input	Sample output
4 **2 1 4 3 3 1 4 2**	**1**

Sample input	Sample output
3 **1 3 2 1 3 2**	**3**

Sample input	Sample output
3 **1 3 2 2 3 1**	**0**

Southeastern European Regional Programming Contest
Bucharest, Romania - Vinnitsa, Ukraine
October 13, 2012

Problem C
Primes

Input File: C.IN
Output File: standard output
Program Source File: C.C, C.CPP, C.JAVA

Let m and n be two integers, $2 \leq m < n \leq 10000000$. Consider the following set:

$$Prime(m, n) = \{ p \mid p \text{ prime and } m \leq p \leq n \}.$$

Compute the cardinal of the set $Prime(m, n)$.

Input
The input file consists of several tests. The input of each test is represented on a single line in the input file. Any two consecutive tests are separated by an empty line. For each test, the values for m and n are given on the same line, separated by exactly one space.

Output
For each test, the result will be written to standard output on a different line (the tests will have the same order as in the input file). The results of any two consecutive tests will be separated by an empty line. For each test, the result will be the cardinal of the set $Prime(m, n)$.

Sample input	Sample output
2 20 **70 110** **5 150**	**8** **10** **33**

Southeastern European Regional Programming Contest
Bucharest, Romania - Vinnitsa, Ukraine
October 13, 2012

Problem D

Longest Prefix Match

Input File: D.IN
Output File: standard output
Program Source File: D.C, D.CPP, D.JAVA

In the Internet, a router forwards packets based on their destination address by consulting a forwarding table which specifies for each destination which path (if any) the packet should take next. A destination address is an unsigned integer. The forwarding table contains many distinct prefixes (thousands), and each prefix is a tuple (`M,L`) where `M` is an address mask (an unsigned integer) and `L` is a length specifying the number of relevant bits in the mask. A destination address `D` matches a prefix (`M,L`) if the `L` most significant bits of `D` match the `L` most significant bits of `M`.

Longest Prefix Match: whenever a packet arrives, the router must find the longest prefix from its database that matches the destination address of the packet. You are asked to implement Longest Prefix Match for a router that has to forward a `few million` packets as quickly as possible.

On the first line, the input gives two integers `X` and `Y`. `X` denotes the number of prefixes in the router's forwarding table, and `Y` gives the number of packets to be forwarded. The next `X` lines describe the prefixes, each line describing one prefix by the mask (hexadecimal) followed by the length (base`10`). Each prefix is identified by number the line it appears on minus one (i.e. the first prefix has id `0`, the second `1`, etc). The next `Y` lines give the destination addresses of the packets to be forwarded, with one packet on each line (the addresses are also given in hexadecimal). The output contains `Y` lines. Line `k` represents the outcome of the longest prefix match algorithm for the `k`'th address in the input file. The outcome is either the identifier of the longest matching prefix or `-1` if no matching prefix was found.

Sample input	Sample output
5 5 **0xFFFFFF00 24** **0xFF000000 8** **0xAF230000 16** **0x3 31** **0 0** **0xFFFF0000** **0xF0FF0000** **0xFFFFFF00** **0xAF320000** **0x2**	**1** **4** **0** **4** **3**

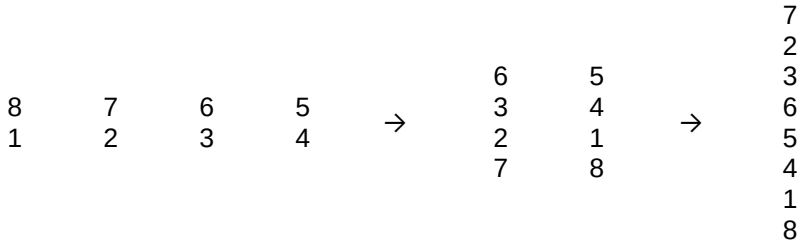
Problem E
Stripe

Input File: E.IN
Output File: standard output
Program Source File: E.C, E.CPP, E.JAVA

Let K be some positive integer. We have a paper stripe with $N=2^K$ cells. Cells are numbered left to right with numbers from 1 to N. Here is how the stripe looks like initially (this is a side view, not a top view!) for K=3, and correspondingly, N=8:

1 2 3 4 5 6 7 8

We can fold the stripe in the following way: stripe is marked in its center and, keeping one half not touched, fold the other up or down. First, we fold right half up. Then, we fold left half down, and finally, we fold left half up. This is how sample stripe will look like after each move:



As we see, numbers on the stripe from top to bottom stand in the following order: 7, 2, 3, 6, 5, 4, 1, 8.

This is what we need to find. K in this task is fixed and equals 21. So we have a stripe with 2^{21} cells; this stripe is folded 21 times, according to a sequence of folding instructions: LU (we bind left half up), LD (we bind left half down), RU (we bind right half up) or RD (we bind right half down). We need to determine the order of the numbers on the folded stripe. More exactly, we need to know 1024 numbers – they should appear on output.

Input data. Input file contains the folding instructions and consists of 21 lines – each line contains two symbols: LU, LD, RU, or RD. The M-th line contains the description of the M-th (1≤M≤21) fold.
Output data. Output consists of exactly 1024 lines. The M-th line (1≤M≤1024) contains the M-th number from top.

Sample data are too big to show on paper, but if K=3 and you have to output the first 5 numbers, the output is the following:

Sample input	Sample output
RU **LD** **LU**	**7** **2** **3** **6** **5**

Problem F

Maximal Number of Divisors

Input File: F.IN
Output File: standard output
Program Source File: F.C, F.CPP, F.JAVA

Help Mr. Strange to solve the following problem: "Find the maximum number of positive divisors of all integers in the interval from 1 up to and including N ($1 \le N \le 42^{42}$)". Since it is Mr. Strange who solves the problem, the situation is complicated by the fact that he dislikes number P. Therefore, you should not find the actual maximum number of divisors of all possible numbers from 1 to N, but the maximum number of divisors only for numbers from 1 to N, which are not divisible by P.

The input file contains several data sets. Your program reads the number of data sets T ($2 \le T \le 42$), followed by the actual data sets. Each data set contains exactly three lines. The first line contains the number P (which is disliked by Mr. Strange; $2 \le P \le 1{,}000{,}000{,}007$). The second line contains the number K ($1 \le K \le 42$) of intervals [1, N] for which the solution is to be found. The third line contains K space-separated integers, N_i , each in the range $1 \le N_i \le 42^{42}$ representing the upper limit of each interval

The result for each of the T data sets should be written on separate lines. The results for the intervals [1, N_i] should be space-separated.

Sample input	Sample output
2 **13** **2** **8 42** **6** **2** **8 42**	**4 9** **4 8**

Explanation. In each data set you are asked to find the maximal number of divisors for numbers in intervals [1, 8] and [1, 42]. If Mr. Strange dislikes P=13, the maximum number of divisors is 4 and is reached for the numbers 6 (divisors 1, 2, 3 and 6) and 8 (divisors 1, 2, 4 and 8); the maximum number of divisors 9 is reached for the number 36. If Mr. Strange dislikes P=6, numbers 6, 12, 18, 24, 30, 36, 42 are prohibited. So, the same maximum 4 is reached only for the number 8; no number in the interval [1, 42] has 9 divisors, but the maximum number 8 is reached for the number 40.

Problem G

Miraculous Drug

Input File: G.IN
Output File: standard output
Program Source File: G.C, G.CPP, G.JAVA

Joe is an enthusiast biomedical researcher. He is very close to discover the cure for a terrible disease. In order to prepare the miraculous drug he needs to buy a special enzyme, that is quite expensive and unfortunately loses its properties after a fixed period of time. Now Joe is in the clinical trial phase. He needs a drug available at each hour. Thus he has to prepare exactly the same quantity of drug every hour. The price of the enzyme might vary from hour to hour. The cost of the enzyme on hour `i` is c_i. The life time of the enzyme is h hours. Given the prices for the next n hours, Joe has to find out the optimal cost to purchase enzymes such that the drug is available in each of the n hours. If the price is the same Joe prefers to buy fresh enzymes, not to stock them. We assume an unlimited quantity of enzymes is available each hour. Can you help Joe?

The program input is from a text file. The file contains several data sets. A data set starts with the number n ($n<10000$) of hours. Follows h ($h<10000$) the number of hours of the enzyme life time, b the starting point, e the ending point of the printing interval ($1 \le b, e \le n$), and the enzyme costs c_i ($c_i<10000$), $i=1..n$. The program prints for each hour in the interval $[b,e]$ the number of enzymes Joe decides to buy.

White spaces can occur freely in the input. The input data are correct and terminate with an end of file. For each set of data the program prints the result to the standard output from the beginning of a line. An input/output sample is in the table below. There are two data sets. For the first one n=6, h=3, b=1, e=6, and the costs are 5 4 4 3 5 6. The result consists of the numbers of enzymes bought every hour, printed from the beginning of the line, separated with tabs.

Sample input	Sample output
6 3 1 6 **5 4 4 3 5 6** **3 3 2 3** **9000 9000 9000**	**1 1 1 3 0 0** **1 1**

Problem H
Softville

Input File: H.IN
Output File: standard output
Program Source File: H.C, H.CPP, H.JAVA

It's common practice in the village of Softville to produce software in a specific way: in the first day the programmer writes several lines of code (at least one line) and every next day he or she writes one line more than the day before.

After they turn 10 years old, every citizen of Softville should be initiated by writing from scratch a very complicated program. The code of the program is the same for all citizens and contains exactly *N* lines. It takes a different number of days for each citizen to complete the task. Nevertheless the rule always remains true: every day each programmer writes one line more than he wrote the previous day.

You'll be given data about how many days it took citizens of Softville to write their initial program. You should determine the least possible value for *N* i.e. the number of lines of the program.

Input data. The first line of the input file contains the number *m* ($1 \le m \le 1000$) of citizens of Softville. The second line contains *m* numbers which represent the number of days each citizen was occupied writing the initial program. All numbers in the second line are positive integers less than 2^{63}.

Output data. The only number in the output should be the least possible value for *N*. In case input data wouldn't be consistent regardless of value of *N* you should output 0, otherwise it's guaranteed that the least possible value for *N* is less than 2^{63}.

Sample input	Sample output
2 **2 3**	**9**

Sample input	Sample output
2 **4 2**	**0**

Problem I
Tree

Input File: I.IN
Output File: standard output
Program Source File: I.C, I.CPP, I.JAVA

A weighted tree is a tree where each edge is labeled with a number representing the edge's length. All lengths are positive. For each node, you have to find the maximum possible distance to any other node in the tree.

Input data. The input file contains the description of the tree. The first line of the input file contains one integer N, 2<=N<=50000. Each of the following (N-1) lines contains the description of the tree's edges. Each edge is described by three positive integers. The first two integers are the labels of the nodes connected by this edge, ranging from 1 to N, the third number – the length of the edge. The total length of all edges does not exceed 2^{31}-1. It is guaranteed that the file contains a correct description of the tree.

Output data. The output consists of exactly N lines: the k-th line contains the distance from node k (k=1..N) to the most distant node.

Sample input	Sample output
6 **1 5 3** **2 6 3** **6 1 1** **1 3 5** **4 6 4**	**5** **9** **10** **10** **8** **6**

Problem J
VAN DINSKY

Input File: J.IN
Output File: standard output
Program Source File: J.C, J.CPP, J.JAVA

You must help would-be painter Vincent van Dinsky mix colors for his paintings. His master gave him a book of color mixing rules plus a color palette, and told him to produce the set of colors that are required for a painting, by experimenting until he finds the minimum number of color mixes that is required for each color in the painting.

Notes:

- all colors are named using the characters `a-z0-9` (lowercase)
- a color mixing rule is made by three colors written on the same line, meaning "color1 mixed with color2 gives color3" – e.g. "`yellow cyan green`"
- mr. van Dinsky is not willing to do random color mixing experiments – if he doesn't find in the book what is the result of mixing `yellow` with `green`, he will never attempt to do this operation. He is also very bad at logic inference – e.g. if the book says that yellow+cyan=green, yellow+magenta=red and red+cyan=black, he will not assume that green+magenta=black (unless the book explicitly tells what is the result of mixing green & magenta). On the other hand – he is fully aware that it doesn't matter what color you start with – mixing yellow & green is the same as mixing green & yellow.

For an input that contains, in order

- all the color mixing rules in the book
- an empty line
- one or several data sets (painting tasks), made of two lines each:
    - all the colors available initially on the palette
    - all the colors required for the painting

you should produce an output that contains one line for each painting (i.e. each data set); on that line, for each color in the painting, the minimum number of color mixes that will produce the desired color (starting from the colors available initially on the palette; the result is '`-1`' if it's impossible to obtain the desired color)

You can assume that the input is correct, there are less than `50000` color mixing rules in the rule book and less than `1000` colors

Sample input	Sample output
cyan yellow green **cyan magenta blue** **yellow magenta red** **red green black** **<done/>** **cyan yellow magenta pink** **cyan black pink red brown** **cyan yellow red** **blue black**	**0 3 0 1 -1** **-1 2**