Problem A. Business

Input file:	`business.in`
Output file:	`business.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

It is hard to do business nowadays. Large enterprises tend to depend on many local small firms and enterprises which are often unprofitable. On the other hand, big ones could lose even more money because of corruption. It may even turn out that the whole system of enterprises functions at a loss. It should prove useful to detect such situations and decide which enterprises could be shut down without losing profit.

You control a system of enterprises, and you are planning an improvement. The improvement will consist in terminating some enterprises while leaving other ones intact. For each enterprise that your scheme will consider useful, all the enterprises it depends on should also be left intact.

The enterprises are numbered from 1 to N. Each enterprise i in your system has an integer A_i associated with it — the annual profit of running that enterprise. When A_i is negative, it means that the corresponding enterprise functions at a loss. Moreover, you have the graph of immediate dependencies at hand. Your task is now to pick a subset of enterprises that will be running so as to maximize the total annual profit of your system. Your wise manager have however foreseen such an occasion, and simplified the dependency graph in such a way that no two enterprises i and j depend on the same enterprise k, and there are no cyclic chains of dependencies.

Given N and all A_i-s, find out what will be the maximal total annual profit of your system after improvement.

Input

There are two integers N and M ($1 \leqslant N \leqslant 10\,000$, $0 \leqslant M \leqslant 100\,000$) on the first line of the input file. The second line contains N numbers $A_1, A_2, \ldots, A_N$ ($|A_i| \leqslant 10^9$) separated by spaces. The following M lines describe the dependency graph and contain two integers B_j and E_j ($1 \leqslant B_j, E_j \leqslant N$) each meaning that enterprise B_j depends on E_j.

Output

On the first line of the output file you should write a single integer — the maximal total annual profit of your system after you plan and perform an optimal improvement. On the second line, output numbers of the enterprises left operational in any order. Note that it is possible to close all enterprises. If there are multiple solutions, output any of them.

Example

business.in	business.out
4 3 -10 10 -10 10 1 2 2 3 2 4	10 2 3 4

Problem B. Coding

Input file:	`coding.in`
Output file:	`coding.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Coding theory is a branch of computer science which deals with error-prone means of transmitting data across noisy channels. In this problem, your task will be to construct a *code* which is optimal in some sense.

Let the alphabet Σ consist of three Latin letters, `A`, `B`, and `C`. Our *message* will be encoded as a string of length n over Σ. Furthermore, we have a so-called *standard string* S, also of length n, with the following property: only the strings with the same number of `A`'s, `B`'s and `C`'s as the standard string could be transmitted over the channel.

During the transmission process, the following error could occur: the channel takes three positions in the string which contain letters `A`, `B` and `C`, and changes them either into `B`, `C` and `A` or into `C`, `A` and `B`, respectively. For example, introducing a single error could transform the string `AABC` into `BACA` or `ACAB`, but could not transform it into `BAC`, `ACBA` or `BBCA`. Surely, the number of `A`'s, `B`'s and `C`'s remains the same.

We are now interested in constructing a code, which is a subset $\mathcal{C}$ of the set of all transmittable strings $\mathcal{T}$, that would be able to correct at most one error introduced by the channel. It means that for every string $P \in \mathcal{T}$, there is no more than one string $C \in \mathcal{C}$ that could be transformed into P by introducing at most one error. The quality of the code $\mathcal{C}$ is its size $|\mathcal{C}|$, which corresponds to the number of different messages we can represent by a single code.

Given the standard string S, output the code $\mathcal{C}$ of maximal possible size.

Input

The input file consists of a single string S of length n ($1 \leqslant n \leqslant 8$) on a line by itself. S will consist only of the first three uppercase Latin letters.

Output

On the first line of the output file, write $k = |\mathcal{C}|$ — the size of your code. The next k lines should contain k elements of $\mathcal{C}$ in any order. If there are multiple solutions, output any of them.

Example

coding.in	coding.out
ABC	2 ABC BAC

Problem C. Cyclic suffixes

Input file:	`cyclic.in`
Output file:	`cyclic.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Consider a string $S = s_1 s_2 s_3 \dots s_{n-1} s_n$ over an alphabet Σ. A *cyclic expansion* of order m over a string S is the string $s_1 s_2 s_3 \dots s_{n-1} s_n s_1 s_2 \dots$ of m characters; that is, we concatenate instances of S until the resulting string has sufficient length, and then take the first m characters as the result.

Let *cyclic string* $\tilde{S}$ be the infinite cyclic expansion over a string S.

Take a look at the suffixes of a cyclic string $\tilde{S}$. Obviously, there are no more than $|S|$ different ones among them: the $(n+1)$-th one is equal to the first (the cyclic string itself), the $(n+2)$-th is equal to the second, and so on. However, there may be even less different suffixes. For example, if $S =$ `abab`, the first suffixes of $\tilde{S}$ are:

$$\begin{aligned} \tilde{S}_1 &= \texttt{abababab}\dots \\ \tilde{S}_2 &= \texttt{bababababa}\dots \\ \tilde{S}_3 &= \texttt{abababab}\dots \\ \tilde{S}_4 &= \texttt{bababababa}\dots \end{aligned}$$

etc. Here, only two different suffixes can be found, while $|S| = 4$.

Let us order the first $|S|$ suffixes of $\tilde{S}$ lexicographically. If two suffixes are equal, the one with the lower index comes first. We are now interested in the following problem: what is the position of our original string $\tilde{S}$ after sorting?

The following example shows the ordering for $S =$ `cabcab`.

$$\begin{aligned} (1)\quad \tilde{S}_2 &= \texttt{abcabcabca}\dots \\ (2)\quad \tilde{S}_5 &= \texttt{abcabcabca}\dots \\ (3)\quad \tilde{S}_3 &= \texttt{bcabcabcab}\dots \\ (4)\quad \tilde{S}_6 &= \texttt{bcabcabcab}\dots \\ (5)\quad \tilde{S}_1 &= \texttt{cabcabcabc}\dots \\ (6)\quad \tilde{S}_4 &= \texttt{cabcabcabc}\dots \end{aligned}$$

Here, the position of $\tilde{S} = \tilde{S}_1$ is 5.

Given a string S of length n, find the number of $\tilde{S}$ in the specified ordering.

Input

On the first line of the input file, a string S ($1 \leqslant |S| \leqslant 1\,000\,000$) is given. The input consists of lowercase Latin letters only.

Output

Output the only number on a line by itself — the number of $\tilde{S}$ in the specified ordering of its first $|S|$ suffixes.

Examples

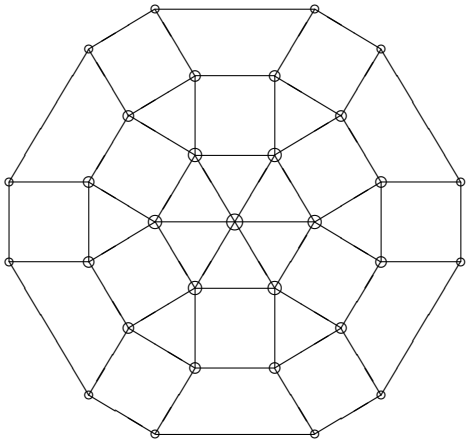
cyclic.in	cyclic.out
abracadabra	3
cabcab	5

Problem D. MagLev

Input file:	`maglev.in`
Output file:	`maglev.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

In a big megapolis far, far away, people use a system of magnetic levitation trains (MagLev for short) to get from point A to point B really fast. Each MagLev station in the megapolis has a rank associated with it. Rank of a station means how much MagLev tracks one could take from that station. The central station has the highest rank, $n \leqslant 10$, and the n tracks connect it to n stations of rank $n-1$. Each station of rank $k < n$ is connected to one station of rank $k+1$, two stations of rank k each and $k-3$ stations of rank $k-1$. Obviously, there are no stations of rank less than 3. The stations of the same rank form a sort of concentric circles; tracks intersect only at endpoints. All tracks are bidirectional.

Below is the picture for $n = 6$; more detailed specification of MagLev system follows.



No two stations of rank k are connected to the same station of rank $k-1$. For each station of rank $k > 3$, all adjacent stations of rank $k-1$ are numbered clockwise starting from zero (it is irrelevant which station will be numbered 0). Thus, the stations adjacent to the central one are numbered from 0 to $n-1$, inclusive; here, each station i of rank $n-1$ is connected to stations $i-1$ and $i+1$ modulo n.

On the other side, for $k < n$, each station of rank $k-1$ is given a number in the range from 0 to $k-4$ according to the clockwise ordering around the corresponding station of rank k. Here, i-th of them is connected to the $(i-1)$-st and $(i+1)$-st ones adjacent to the same station of rank k (of course, if they exist). The two exceptions are stations numbered 0 and $k-4$; for two stations u and v of rank k, station $k-4$ adjacent to u is connected to station 0 adjacent to v if and only if either u and v are adjacent to the same station of rank $k+1$ and $u+1 = v$ or the numbers $u = k-3$, $v = 0$ and u is already connected to v.

Citizens of that far megapolis use an interesting system of path notation. Each track in the path is represented by either a digit or a lowercase English letter. When a track goes to a station of lower rank, it's denoted by a single digit — the number of the lower-ranked station among the ones adjacent to the higher-ranked one. A track between stations of the same rank is denoted as '`l`' when the movement around the station of higher rank is counter-clockwise and '`r`' if it is clockwise. Finally, when the track goes to a station of higher rank, the corresponding symbol is '`b`'.

For simplicity, citizens name stations by strings that represent the paths to them from the central station. However, with such naming, one station could have different names.

John lives near the station named S, and his work place is located at the station named T. Travelling along every track costs the same amount of money. That's why John always takes the route with the least number of tracks when travelling by MagLev. However, he is bored travelling along the same route over and over again. That is why he asked you to find how many different routes of minimal length between stations S and T exist.

Input

There is a single integer N, $4 \leqslant N \leqslant 10$, on the first line of the output file. On the following two lines, you are given two strings, S and T, one string on a line. The strings are valid station names, they are non-empty, do not point to the same station, and their lengths do not exceed 10 000.

Output

Output a single integer on a line by itself — the required number of routes.

Example

maglev.in	maglev.out
6 210 210llllb	8

Problem E. Quadrilaterals

Input file:	quadrilaterals.in
Output file:	quadrilaterals.out
Time limit:	3 seconds
Memory limit:	64 megabytes

You are given N points in general position on a plane, that is, no two points coincide, and no three ones lie on a same straight line. Find out how many different convex quadrilaterals with vertices in these points could be constructed.

Input

There is a single integer N ($4 \leqslant N \leqslant 1\,500$) on the first line of the input file. The following N lines contain two integers X_i and Y_i each — the coordinates of the points. All coordinates do not exceed 10^8 by absolute value.

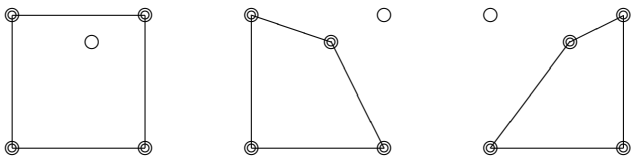
Output

Write a single integer on a line by itself — the number of different convex quadrilaterals that could be constructed.

Examples

quadrilaterals.in	quadrilaterals.out
4 -1 -1 -1 1 1 1 1 -1	1
5 0 0 5 0 3 4 0 5 5 5	3

The last example is illustrated below.



Problem F. Impudence Queue

Input file:	queue.in
Output file:	queue.out
Time limit:	2 seconds
Memory limit:	64 megabytes

A group of n workers go to the canteen for a dinner. They arrive sequentially and form a queue. Eventually, a worker at the head of the queue gets served, and the workers left in the queue move one step closer to the serving point. However, when a worker arrives, he does not always go and stand at the tail of the queue — sometimes he sees a friend in the middle of the queue and passes directly to him. More formally, each worker p is characterized by his *impudence* I_p, and some pairs of workers are friends (the friendship relation is mutual). When a worker p arrives, he searches the queue starting from its head until he finds another worker q such that

- p and q are friends,
- for every worker r between q and the tail of the queue, $I_p + I_q > 2 \cdot I_r$.

If p finds such q, he stands right before him if $I_p > I_q$ and right after him otherwise, moving the whole tail of the queue one step farther from its head. If p does not find any such q, he just stands at the tail of the queue and does not move anyone.

Given a sequence of arrivals and servings, find out the order in which workers get served.

Input

There are two integers, n ($1 \leqslant n \leqslant 10\,000$) and m ($1 \leqslant m \leqslant 100\,000$) on the first line of the output file. The second line contains n integers $I_1, I_2, \ldots, I_n$ ($1 \leqslant I_p \leqslant 10^9$) separated by spaces — impudences of workers in the order they arrive to the canteen. The next m lines describe friendship relations and contain two distinct numbers p_k and q_k each meaning that the workers p_k and q_k are friends.

The last line contains $2n$ binary digits separated by spaces and describes the sequence of arrivals and servings. The number 1 means that the next worker have arrived, while the number 0 means that the worker at the head of the queue got served. This line contains exactly n ones and n zeroes. Workers arrive in the same order as they are given in the second line. An empty queue never gets served.

Output

Output n integers on the first line of the output file — the order in which the workers got served.

Example

queue.in	queue.out
4 4 2 3 5 1 1 2 1 3 2 3 3 4 1 1 1 1 0 0 0 0	3 2 1 4

Problem G. Raccoons

Input file:	raccoons.in
Output file:	raccoons.out
Time limit:	2 seconds
Memory limit:	64 megabytes

N raccoons live in the forest. They want to become acquainted with each other, because winter will come soon, and they want to share a common large hollow — the only warm place in the forest. Unfortunately, all of them are rather shy, so it takes much time for them to make themselves comfortable in company. In fact, they are so shy that a set $\mathcal{A}$ of K raccoons could get acquainted with each other in $f(K)$ seconds; to make things worse, other $N-K$ raccoons will not be able to communicate with each other during that period. For simplicity, let $f(K)$ be a polynomial of degree $M \leqslant 10$ with nonnegative integer coefficients. But what's really terrific is that the set $\mathcal{A}$ can not get acquainted before all K subsets $\mathcal{B} \subset \mathcal{A}$ such that $|\mathcal{B}| = |\mathcal{A}| - 1$ get acquainted.

You could easily count the time it would take for all the subsets of raccoons to get acquainted: it's the sum $S = \binom{N}{1} f(1) + \binom{N}{2} f(2) + \ldots + \binom{N}{N} f(N)$ seconds. Note that it takes $f(1)$ seconds even for a single raccoon to get enough confidence to be able to communicate with others; on the other hand, empty set of raccoons does not need to be taken into consideration.

However, the frost could strike earlier. More specifically, for each second i, $1 \leqslant i \leqslant S$, the probability that the cold starts during that second is equal to $\frac{1}{S}$. When the frost strikes, all raccoons should go into the same hollow and spend all the winter there. But if they are not enough acquainted with each other, they would be too shy to gather together, and some of them would rather freeze out than come and share the hollow.

Fortunately, all is not lost yet. Consider all N subsets $\mathcal{D}_i$ such that $|\mathcal{D}_i| = N - 1$. If R of them already got acquainted, the probability that all the raccoons successfully share the hollow is $\frac{R}{N}$. Thus, if none of $\mathcal{D}_i$ got acquainted, the probability is zero; conversely, if all of them got acquainted, there is no need to wait and get acquainted in the whole set, because the raccoons are already communicative enough to share the hollow.

As you can see, the order in which the sets get acquainted is of vital importance to the raccoon population of the forest. Help raccoons to devise the order which maximizes the average probability of the event that all raccoons will survive, and output that order.

Input

There are two integers, N and M ($2 \leqslant N \leqslant 16$, $1 \leqslant M \leqslant 10$) on the first line of the input file. The second line contains $M + 1$ integers a_M, a_{M-1}, … a_1, a_0 — the coefficients of $f(x) = a_M x^M + \ldots + a_1 x + a_0$ ($0 \leqslant a_i \leqslant 1\,000$). The first one, a_M, is greater than zero.

Output

Output $2^N - 2$ numbers — the binary representations of the subsets in the order they get acquainted. Each representation is a string of N binary digits. The set of all raccoons should not be present. If there are multiple solutions, output any of them. Adhere to the sample output below.

Example

raccoons.in	raccoons.out
2 1 1 0	10 01

Here, $f(x) = x$, and $S = 1 \cdot 2 + 2 = 4$. Firstly, the first raccoon gets enough confidence. Secondly, the second one does the same. The average probability is $\frac{1}{4}(0.5 + 1 + 1 + 1) = 0.875$.

Problem H. Robots' DNA

Input file:	`robots.in`
Output file:	`robots.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

New inventions in DNA building technology allowed to conduct the greatest experiment in building biological robots. It was done by scientists from Research Institute of Given Strings (RIGS).

The DNA of these robots was selected to have $M = 2^n$ characters for some $n \geq 2$. This was done for simplifying the management software. Moreover, for technical reasons the Robots' DNA is not an ordinary string, but a cyclic one, so it can be read from an arbitrary position.

One interesting thing to be investigated during this experiment is the behaviour of Robots' mutations. After a long time, there have been found some different kinds of robots. To reconstruct the mutation tree, scientists from RIGS need to solve one particular task. Namely, they need to compute the *similarity coefficient* of two Robots' DNAs. The similarity coefficient is the number of matching letters in the best alignment of two given DNAs. The more letters match, the better is alignment.

You need to write a program that will find the best alignment.

Input

The first line of the input contains one integer M ($4 \leq M \leq 131\,072$). The next two lines are the DNAs to be investigated. Both of them consist of exactly M characters from the set '`A`', '`C`', '`G`' and '`T`'.

Output

You must output the value of the best alignment and the optimal shift. The value is the number of matching characters, and the shift is the (non-zero) number of terminating characters of second DNA to be appended to the beginning of it to get the best alignment.

Example

robots.in	robots.out
16 ACGTACGTACGTACGT CGTACGTACGTACGTC	15 1

Problem I. Sums

Input file:	sums.in
Output file:	sums.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Nick and his younger brother Pete play the following game. There are N natural numbers written on N cards, one number on a card. First, Nick selects K cards with numbers. Then, Pete gives Nick another natural number, P. Nick is to show Pete some cards so that the sum of the numbers on them is P. If he does that, he wins; otherwise, the winner is Pete.

As an example, suppose that the numbers on the cards are 1, 2, and 3, $K = 2$, and Nick chooses cards with numbers 1 and 3 written on them. Now, if Pete gives the number $P = 4$, Nick is able to represent it as a sum of the numbers on both selected cards, while if $P = 2$, he is unable to show Pete a set of cards with such a sum of numbers on them.

The game does not sound too difficult for Pete — he could just give such a large P that the sum of numbers on K cards is less than P no matter which cards Nick chooses. However, Pete is too young, and he didn't yet learn to operate large numbers. So, Nick's strategy is to choose such a set of K cards that the minimal natural number which could not be presented as a sum of numbers on some of these K cards would be maximal possible.

However, the set of cards is too large, and Nick feels unable to choose an optimal set. Help him to do it!

Input

There are two integer numbers, N and K ($1 \leqslant K \leqslant N \leqslant 10\,000$) on the first line of the input file. The second line consists of N integers $A_1, A_2, \ldots, A_N$ ($1 \leqslant A_i \leqslant 10^9$) — the numbers written on the cards.

Output

You should output K numbers on the second line of the output file. These should be the numbers on the cards selected by Nick. If there are multiple solutions, output any of them. Separate numbers by single spaces.

Examples

sums.in	sums.out
3 2 1 2 3	1 2
4 3 1 2 2 3	1 2 3

Problem J. Psychic Test

Input file:	`test.in`
Output file:	`test.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

You are going to attend the Psychics Congress — a congress for people who claim to possess certain kinds of extra-sensory perception. You are unsure of whether you possess such abilities; however, you've got the knowledge of what the first test for entrants will be. Here is the procedure.

Two entrants sit in front of each other, both holding a sheet of paper. The guru — the one accredited to perform the test — gives them a natural number N. Then, entrants look deep into the eyes of each other for one minute. After that, they turn around so that they don't see each other and write numbers on their sheets of paper: the first entrant writes a number A ($0 \leqslant A < N$) and the second writes B ($0 \leqslant B < N$). Finally, the entrants perform the following calculations.

Entrants start with both pronouncing a number, *"one"*. Next, first entrant adds A to his last pronounced number, takes the remainder modulo N and pronounces the result. Simultaneously, second entrant multiplies his last pronounced number by B, also takes the remainder modulo N and pronounces the number he got. If the numbers differ or they are the same, but that number was already pronounced, the test ends. Otherwise, they repeat the step of adding and multiplying until one of the end conditions above happens.

The purpose of the test is to measure the level of mutual understanding which the entrants could achieve. The more numbers the entrants pronounce, the better is the level. The guru chooses pairs of entrants at random (or maybe for some unknown reason?) so you can not cheat and arrange the numbers A and B for every possible N beforehand. So, your only hope is that your entrant will be clever enough to pick such a number A or B that you could pick a corresponding one so that the resulting sequence you both pronounce will have the maximal possible length. Furthermore, if there will be many such numbers, it's natural to choose them so that the pronounced sequence will be lexicographically the least.

You will only have one minute to devise such a pair of numbers, so you decide to write a simple program to train your skill and gain an understanding of what they look like.

Given N, find a pair of numbers A and B such that the pronounced sequence of numbers will be lexicographically minimal among the sequences of maximal length, and output that sequence.

Input

The input consists of a single integer N, $4 \leqslant N \leqslant 10^9$, on a line by itself.

Output

On the first line of output, write an integer K which is the length of your sequence. On the second line, write K numbers separated by spaces — the elements of the pronounced sequence in the order of pronunciation. Note that the first of them will always be 1.

Examples

test.in	test.out
16	4 1 5 9 13
5	2 1 0