Problem A. Document

Input file:	`document.in`
Output file:	`document.out`
Time limit:	1 second
Memory limit:	256 megabytes
Detailed Feedback:	**full**

It is necessary to produce legislative framework document, just to be able to organize an important event. To be entered into force, this document must be signed by N officials, and the order in which officials sign the document is fixed. We assume that officials are numbered in the order of signing, starting from the official number 1, and ending with official number N.

It is known that every official accept documents only at fixed set of days of weeks (visiting days). Also, despite the week has seven days, officials accept documents only on the working days (Monday to Friday). Every official has at least one visiting day within week.

One official needs exactly one day to sign the document, i.e. if an official reviews the document at some day, other officials can not review this document that day.

Some Monday morning you get a document to be signed (let us count this day as the first day). Write a program which will calculate the minimum number of days required to get all signs for this document in the specified order. If the document can not be completed within one week, you can continue to sign it next week, and so on.

It is guaranteed that the document is very well prepared and any official will sign it at the first submit within one day, without any claims.

Input

First line of input file contains N — the number of officials ($1 \leq N \leq 50$). The following N lines contain five space-separated integers each, corresponding to the days of week from Monday to Friday, inclusive. The number in i-th row, corresponding to j-th day of week, is equal to `0`, if i-th official doesn't accept documents at j-th day of week, otherwise it equals `1`.

Output

Output file contains one integer — the minimal number of days required to sign the document.

Examples

document.in	document.out

`2`
`1 0 1 0 1`
`1 0 0 1 0`
	`4`

`2`
`0 1 0 0 1`
`1 1 0 0 0`
	`8`

Examples Explanation

In the first example, the document must be signed by two officials — the first one accept documents on Mondays, Wednesdays and Fridays, and the second one — on Mondays and Thursdays. The fastest way to sign the document is one of the following: you submit the document to the first official on Monday or on Wednesday, and then submit it to the second official on Thursday, so you will need `4` days to sign document. In the second example, the first official works on Tuesdays and Fridays, and the second one — on Mondays and Tuesdays. Even if you submit the document to the first official on Tuesday, you will not complete the process within first week, you can only do this on the next Monday, and the whole process will take `8` days.

Problem B. Frodo and the Monster

Input file:	`monster.in`
Output file:	`monster.out`
Time limit:	1 second
Memory limit:	256 megabytes
Detailed Feedback:	none

Frodo is fighting with the monster, whose heads are numbered sequentially from 1 to K. Monster is cybernetic genome carrier and it's characterised with unusual properties.

If Frodo cuts a head with an odd number X, then Y new heads will appear on this place, where Y is the maximal prime number less than X. If Frodo cuts the first head, no new head will appear. It's known, that the monster's neck can not accommodate more than 30 000 000 heads (it means, that if sum of Y and the count of remaining heads is greater than 30 000 000, then the count of heads will be exactly 30 000 000).

If Frodo cuts a head with an even number Z, the monster will lose all heads with numbers containing the same number of ones in binary notation as Z. If monster loses all heads, then the fight ends.

After each Frodo's strike (after appearance of new heads as well as after losing of some heads) monster's heads are renumbered starting with 1.

Frodo doesn't have time to examine whether numbers of monster's heads are odd or even and he cuts them unsystematically. Fortunately, he has modern laser sword which memorises numbers of cut heads in the moment of the cut.

The fight was finished in the evening, but tired Frodo cannot count the heads left.

You need to write a program which counts the number of heads after the fight is finished.

Input

In the first line of the input file there are two space separated integer numbers K ($2 \le K \le 30\,000\,000$) and N ($2 \le N \le 200\,000$), where K is the initial count of monster heads, and N is the number of cuts. Each of the next N lines contains an integer number. These numbers describe the sequence of cutting heads. Head numbers are correct, i. e. they never exceed the total count of monster heads at the moment of the strike.

Output

Write to the output file the count of heads left on the monster's neck after the fight is finished.

Example

monster.in	monster.out

7 4
5
9
6
4
	5

Example Explanation

After cutting 5th head, 3 new heads will appear and the total number becomes 9. After cutting 9th head, 7 new heads will appear and the total number becomes 15. After cutting 6th head, monster will also lose heads with numbers 3, 5, 9, 10, 12. And after cutting 4th head, it will lose heads 8, 2, 1. After the fight monster will have 5 heads left.

Problem C. Cache

Input file:	`cache.in`
Output file:	`cache.out`
Time limit:	1 second
Memory limit:	256 megabytes
Detailed Feedback:	partial

Let's consider a model of some system where a set of N objects of different sizes is being used. An object can be used by the system only when it is in the cache. If the object is not in the cache, it must be put there before use (some other objects may be deleted before this to free enough space). It is enough to have at least S_i units of free space in total in the cache to be able to put the object of size S_i there. Every object has its own caching cost. The cost of deleting object from cache is zero. Cache can contain a subset of objects with total size not more than C. You know the order in which objects will be used by the system. Determine which objects should be deleted from the cache and when they should be deleted to achieve minimal total cost of putting objects into the cache. Initially the cache is empty. Object cannot be put into cache before it will be requested.

Input

The first line of the input file contains three integer numbers: N, C and K ($1 \le N \le 18$, $1 \le C \le 10^9$, $1 \le K \le 100$), where K is the number of requests to put an object into cache. The second line contains N integer numbers S_i, where S_i is the size of the object number i (the size is between 1 and C). The third line also contains N integers: i-th of them is the cost of putting the object number i into the cache (the cost is between 0 and 10^6). The fourth line contains K integer numbers between 1 and N — objects in the order of using them. Numbers in lines are separated by spaces.

Output

The first line of the output file must contain one integer number — the minimal total cost of putting objects into the cache. After that output K lines. i-th of them must contain list of object that should be deleted from the cache before using i-th object from the list. First number in the line, K, is the count of objects being deleted, then K numbers should follow — numbers of objects themselves. Numbers in lines should be separated by spaces.

Examples

`cache.in`	`cache.out`

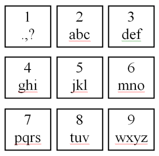
`2 10 3`
`9 8`
`2 1`
`1 2 1`
	`5`
	`0`
	`1 1`
	`1 2`

`2 10 3`
`1 3`
`2 1`
`1 2 1`
	`3`
	`0`
	`0`
	`0`

Problem D. T9

Input file:	`t9.in`
Output file:	`t9.out`
Time limit:	1 second
Memory limit:	256 megabytes
Detailed Feedback:	partial

Many mobile phone users use T9 mode while typing SMS. For example, they type the messages in English in the following way. To type a word by letters, the button corresponding to that letter is pressed only once, no matter how many letters correspond to this button, and no matter which is the position of the letter on this button (see figure), and the software in the phone chooses a word corresponding to the button sequence from the dictionary. If there are several suitable words, then the most frequently occuring word will be offerred first (initially the words with equal frequency are ordered alphabetically).



If the word is wrong, the user presses button '`*`' and the software offers the next word by frequency also suitable for the pressed buttons combination, starting from the next word with the same frequency, if any. If it is also wrong, the button '`*`' will be pressed again etc. For the simplicity we will assume that modern phones contain the full dictionary of the needed words, and the neccessary word will always be found. When the offered word is correct, the user may press "space", may press the button '`1`' which corresponds to the punctuation mark or may finish the typing. When the punctuation mark is wrong, the button '`*`' is pressed also, until the correct character appears. After typing a space or a punctuation mark, the user is allowed to type another space/punctuation mark, is allowed to finish the typing or to start typing the next word. We will assume that three punctuation marks are enough and they are offered in the following order: point ('`.`'), comma ('`,`'), question mark ('`?`').

After the user accepts the word (by pressing space or '`1`'), its frequency in the dictionary is increased by 1 and the new frequency value will be taken into account during typing further words of the message. Also, this word will be the first offered word among all the words of the same frequency, but the order of other words will not be changed. When another word with the same frequency appears, then it will be offered first among the words of this frequency, not changing the order of other words, etc.

You need to write a program which, given a dictionary with initial frequencies of the words and given a sequence of button presses, will write the message that appears on the screen.

Input

The first line of the input file contains an integer N ($3 \le N \le 50\,000$) — the number of words in the dictionary. Each of the next N lines contains one dictionary word and an integer F ($1 \le F \le 1000$) — the initial value of this word frequency (larger value corresponds to larger frequency). The value of the frequency is separated by exactly one space character from the word itself. Dictionary words contain only small english letters. The words in the input file are ordered alphabetically. The word length does not exceed 20 characters, all words are non-empty and different.

Last line of the input file contains string, denoting the sequence of pressed buttons when message is typed, which consists of digits from `1` to `9`, and characters "space" and '`*`'. Length of this line does not exceed 100 000 characters.

Output

Output the text of SMS.

Examples

t9.in	t9.out
5 ad 2 be 1 not 10 or 5 to 50 86 23* 67 668 86 231**	to be or not to be?
3 act 1 bat 1 cat 1 228* 228** 228** 228**1	bat cat act bat.

Examples Explanation

In the second example, at first the software will propose the word "`act`", then "`bat`", which will be accepted by the user. Then the frequency of the word "`bat`" becomes 2. Thus, when the same combination of digits is entered again, the first proposed word is "`bat`", then "`act`" and then "`cat`". Third time the words will be proposed in the following order: "`cat`" (as the new word with frequency 2), "`bat`" (frequency 2), "`act`" (and its frequency becomes equal to the frequency of other words), and in the last case the words will be proposed in order "`act`", "`cat`", "`bat`".