Ancient Manuscript (Latvia)

TASK

Baltic archaeologists are currently engaged in a very important project and have recently found an ancient manuscript that seems to be crucial for the understanding of the culture that inhabited the area they are exploring. The manuscript is full of drawings, so scientists are able to get a general feel for the subject of the document.

However, there is also a written part, and with that scientists are in trouble. Apart from the language used in the writing being a very ancient, several parts of manuscript were destroyed, some letters disappeared, and they are unable to completely understand what is written there.

One of the scientists said, that the words in the manuscript remind him of a language about which it is known that in any word there may be no more than V_C and C_C consecutive vowels and consonants, respectively, and that no more than V_E and C_E consecutive vowels and consonants, respectively, may be equal.

That scientist left the group in search of a more precise information. The others, while waiting for that scientist to return, decided to check whether nothing in the manuscript contradicts his hypothesis and estimate the amount of work that may lie ahead, so they want to know in how many different ways the manuscript can possibly be deciphered. We must help them!

Note: vowels are “aeiou” and there are 21 other letters in the alphabet – consonants.

INPUT

The input file name is `ANCIENT.IN` The first line of the input file contains four integers V_E, V_C, C_E and C_C ($1 \le V_E \le V_C \le 4$, $1 \le C_E \le C_C \le 4$) separated by single spaces. The second line contains one word extracted from the manuscript consisting of up to 15 Latin alphabet lowercase letters with missing characters (if any) designated by “*”.

OUTPUT

The output file name is `ANCIENT.OUT`. One integer, describing in how many ways it is possible to make up a legal word based only on the constraints given. You may assume that the answer will fit into a 64-bit signed integer. It may happen that scientist’s conjecture about the language is incorrect and that there are no ways to make up a legal word; in this case, the answer is, obviously, 0

EXAMPLES

INPUT	**OUTPUT**
1 1 1 1 a**	105
1 1 1 1 b*i	0
1 2 1 2 ancient	1
4 4 4 4 man****ipt	261870
2 2 2 2 *boi*	546

Bus Trip (Estonia)

TASK

There are N towns, and M one-way direct bus routes (no intermediate stops) between them. The towns are numbered from 1 to N. A traveler who is located in the town 1 at time 0 needs to arrive in the town P. He will be picked from the bus station at town P exactly at time T. If he arrives earlier he will have to wait.

For each bus route i, we know the source and destination towns s_i and t_i, of course. We also know the departure and arrival times, but only approximately: we know that the bus departs from s_i within the range $[a_i, b_i]$ and arrives at t_i within the range $[c_i, d_i]$ (endpoints included in both cases).

The traveler does not like waiting, and therefore is looking for a travel plan which minimizes the maximal possible waiting time while still guaranteeing that he'll not miss connecting buses (that is, every time he changes buses, the latest possible arrival of the incoming bus must not be later than the earliest possible departure time of the outgoing bus).

When counting waiting time we have to assume the earliest possible arrival time and the latest possible departure time.

Write a program to help the traveler to find a suitable plan.

INPUT

The input file name is `TRIP.IN`. The first line contains the integer numbers N ($1 \le N \le 50{,}000$), M ($1 \le M \le 100{,}000$), P ($1 \le P \le N$), and T ($0 \le T \le 1{,}000{,}000{,}000$).

The following M lines describe the bus routes. Each line contains the integer numbers s_i, t_i, a_i, b_i, c_i, d_i, where s_i and t_i are the source and destination towns of the bus route i, and a_i, b_i, c_i, d_i describe the departure and arrival times as explained above ($1 \le s_i \le N$, $1 \le t_i \le N$, $0 \le a_i \le b_i < c_i \le d_i \le 1{,}000{,}000{,}000$).

OUTPUT

The only line of the output file `TRIP.OUT` should contain the maximal possible total waiting time for the most suitable possible travel plan. If it is not possible to guarantee arrival in town P by time T, the line should contain –1.

EXAMPLES

INPUT
```
3 6 2 100
1 3 10 20 30 40
3 2 32 35 95 95
1 1 1 1 7 8
1 3 8 8 9 9
2 2 98 98 99 99
1 2 0 0 99 101
```

OUTPUT
```
32
```

The most pessimistic case for the optimal travel plan for the above example is as follows:

Time	**Action**
0…1	Wait in town 1
1…7	Take the bus line 3 from town 1 to town 1
7…8	Wait in town 1
8…9	Take the bus line 4 from town 1 to town 3
9…35	Wait in town 3
35…95	Take the bus line 2 from town 3 to town 2
95…98	Wait in town 2
98…99	Take the bus line 5 from town 2 to town 2
99…100	Wait in town 2

Total waiting time: 1+1+26+3+1=32

INPUT
```
3 2 2 100
1 3 0 0 49 51
3 2 50 51 100 100
```

OUTPUT
```
-1
```

Polygon (Estonia)

TASK

Write a program to find a convex polygon whose sides have the given lengths.

In this task, we consider a polygon to be convex if all its inner angles are strictly greater than 0 degrees and strictly less than 180 degrees.

INPUT

The input file name is `POLY.IN`. The first line of the file contains an integer N, the number of vertices of the polygon ($3 \le N \le 1000$). Each of the following N lines contains an integer a_i, the length of one side of the polygon ($1 \le a_i \le 10{,}000$).

OUTPUT

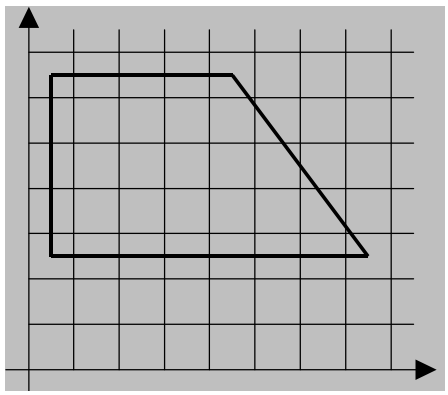
If the desired polygon can be constructed, the output file `POLY.OUT` should contain exactly N lines. Each line should contain two real numbers x_i and y_i ($|x_i| \le 10{,}000{,}000$, $|y_i| \le 10{,}000{,}000$) such that by connecting the points (x_i, y_i) and (x_{i+1}, y_{i+1}) for all $1 \le i < N$ and additionally the points (x_N, y_N) and (x_1, y_1) with line segments, we obtain a convex polygon. The lengths of the line segments must be equal to the numbers given in the input file, but not necessarily in the same order.

The vertices of the constructed polygon can be listed either clockwise or counterclockwise.

If the polygon cannot be constructed, print `NO SOLUTION` on the single line of the output file.

EXAMPLE

INPUT	**OUTPUT**
4	0.5 2.5
7	7.5 2.5
4	4.5 6.5
5	0.5 6.5
4	



GRADING

The grading program considers two lengths equal if they differ by less than 0.001.
Any standard floating point format is acceptable.