2010 Canadian Computing Competition
Day 2, Question 1

Computer Purchase Return

Input: from standard input
Output: to standard output
Source file: `purchase.c`

After considering to buy a brand new Atari or Commodore computer (based on your extensive research in late February), you decide to get the best value for your dollar by building your own.

The computer that you are going to build is composed of T ($1 \leq T \leq 5$) different types of components. Your computer must have exactly one of each type of component.

Each component has an integer cost c_i ($1 \leq c_i \leq 3,000$), an integer value v_i ($1 \leq v_i \leq 3,000$), and type t_i ($1 \leq t_i \leq T$).

Your on-line computer parts store has N different components ($1 \leq N \leq 1,000$) that you can select from.

For a given budget B ($1 \leq B \leq 3,000$), maximize the total value of the components in your computer.

If you cannot construct such a computer, you should print -1.

Input Specification

The first line contains T, the number of types of components your computer requires. The next line contains the number N, followed by N lines of three integers, representing c_i, v_i and t_i, each separated by one space. The last line of input is the budget B.

Output Specification

Output the value of the maximum valued computer you can create which costs at most B, or -1 if you cannot construct a computer.

Sample Input

```
2
5
10 6 1
5 7 1
6 10 2
1 5 1
11 11 2
16
```

Output for Sample Input

```
18
```

Description of Output for Sample Input

Notice that picking the components with cost 11 and 5 yields a computer with value 18. No other combination of components has a higher value.

2010 Canadian Computing Competition
Day 2, Question 2

Space Miner

Input: from standard input
Output: to standard output
Source file: `space.c`

There are M ($1 \leq M \leq 1,000$) planets each with v_i ($1 \leq v_i \leq 10,000$) units of resources and radius r_i ($1 \leq r_i \leq 100$).

Starting from $(0, 0, 0)$, you travel in straight lines through N ($1 \leq N \leq 1,000$) waypoints in a specific order.

Whenever you travel within $D + r_i$ ($1 \leq D \leq 50$) units from a planet's center, you can mine the planet using your tractor beam retrieving v_i units of resources. Note that if you are exactly D units from the surface of the planet, you can mine the planet. If your path takes you through a planet, do not worry, since your spaceship can drill through planets, which makes mining even easier. Also note that you cannot mine the same planet on a subsequent visit.

Give the number of minerals that can be mined on your journey.

Hint: you should do all calculations with 64-bit integers.

Input Specification

On the first line of input is the number M, the number of planets. On the next M lines are five integers describing each of the M planets. These integers are $x_i y_i z_i v_i r_i$, where the planet i is at position (x_i, y_i, z_i), (where $-1,000 \leq x_i, y_i, z_i \leq 1,000$) and this planet has v_i resources and radius r_i. On the next line is the number N, the number of waypoints to pass through. Each of the next N lines contains the position of these waypoints, as three integers $x_i y_i z_i$ ($-1,000 \leq x_i, y_i, z_i \leq 1,000$). The last line of input is the number D, the maximum distance from a planet's surface that your ship can be and still mine a planet.

Output Specification

On one line, output the amount of minerals that you can mine on your journey.

Sample Input

```
3
10 0 0 1 1
0 10 0 2 1
0 0 10 4 1
3
8 0 0
0 7 0
0 0 9
1
```

Output for Sample Input

```
5
```

2010 Canadian Computing Competition
Day 2, Question 3
Shuffle

Input: from standard input
Output: to standard output
Source file: `shuffle.c`

You want to keep some secrets, so you invent a simple encryption algorithm.

You will map each uppercase character and underscore to some other uppercase character and underscore. In other words, this is a permutation of the characters, or, to put it another way, you create a 1:1 and onto map from {`'A'`, `'B'`, ..., `'Z'`, `'_'`} to {`'A'`, `'B'`, ..., `'Z'`, `'_'`}.

However, you will repeatedly apply this encryption in an attempt to make your message more secure.

Input Specification

The input will be 29 lines long. The first 27 lines will each contain a single character from the set {`'A'`, `'B'`, ..., `'Z'`, `'_'`}. The first of these lines represents what the character `'A'` maps to, the second of these lines represents what the character `'B'` maps to, and so on, until the 27th line represents what the underscore character maps to.

On the 28th line will be an integer N ($1 \leq N \leq 2,000,000,000$) which represents the number of times this encryption should be applied.

On the 29th line is T, a string of less than 80 characters from the set {`'A'`, `'B'`, ..., `'Z'`, `'_'`}.

Output Specification

On one line, output the string created after shuffling T a total of N times, using the given shuffle permuation.

Sample Input

```
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
```

```
R
S
T
U
V
W
X
Y
Z
_
A
3
I_LOVE_THE_CCC
```

Output for Sample Input

```
LCORYHCWKHCFFF
```