A. Football

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves football very much. One day, as he was watching a football match, he was writing the players' current positions on a piece of paper. To simplify the situation he depicted it as a string consisting of zeroes and ones. A zero corresponds to players of one team; a one corresponds to players of another team. If there are at least 7 players of some team standing one after another, then the situation is considered dangerous. For example, the situation 00100110111111101 is dangerous and 11110111011101 is not. You are given the current situation. Determine whether it is dangerous or not.

Input

The first input line contains a non-empty string consisting of characters "0" and "1", which represents players. The length of the string does not exceed 100 characters. There's at least one player from each team present on the field.

Output

Print "YES" if the situation is dangerous. Otherwise, print "NO".

Examples

input
```
001001
```
output
```
NO
```

input
```
1000000001
```
output
```
YES
```

B. Lucky Numbers (easy)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves lucky numbers. Everybody knows that positive integers are *lucky* if their decimal representation doesn't contain digits other than 4 and 7. For example, numbers 47, 744, 4 are lucky and 5, 17, 467 are not.

Lucky number is *super lucky* if it's decimal representation contains equal amount of digits 4 and 7. For example, numbers 47, 7744, 474477 are super lucky and 4, 744, 467 are not.

One day Petya came across a positive integer n. Help him to find the least super lucky number which is not less than n.

Input

The only line contains a positive integer n ($1 \le n \le 10^9$). This number doesn't have leading zeroes.

Output

Output the least super lucky number that is more than or equal to n.

Please, do not use the %lld specificator to read or write 64-bit integers in C++. It is preferred to use the cin, cout streams or the %I64d specificator.

Examples

input

```
4500
```

output

```
4747
```

input

```
47
```

output

```
47
```

C. Hockey

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves hockey very much. One day, as he was watching a hockey match, he fell asleep. Petya dreamt of being appointed to change a hockey team's name. Thus, Petya was given the original team name w and the collection of forbidden substrings $s_1, s_2, \ldots, s_n$. All those strings consist of uppercase and lowercase Latin letters. String w has the length of $|w|$, its characters are numbered from 1 to $|w|$.

First Petya should find all the occurrences of forbidden substrings in the w string. During the search of substrings the case of letter shouldn't be taken into consideration. That is, strings "aBC" and "ABc" are considered equal.

After that Petya should perform the replacement of all letters covered by the occurrences. More formally: a letter in the position i should be replaced by any other one if for position i in string w there exist pair of indices l, r ($1 \le l \le i \le r \le |w|$) such that substring $w[l \ldots r]$ is contained in the collection $s_1, s_2, \ldots, s_n$, when using case insensitive comparison. During the replacement the letter's case should remain the same. Petya is not allowed to replace the letters that aren't covered by any forbidden substring.

Letter $letter$ (uppercase or lowercase) is considered lucky for the hockey players. That's why Petya should perform the changes so that the $letter$ occurred in the resulting string as many times as possible. Help Petya to find such resulting string. If there are several such strings, find the one that comes first lexicographically.

Note that the process of replacements is not repeated, it occurs only once. That is, if after Petya's replacements the string started to contain new occurrences of bad substrings, Petya pays no attention to them.

Input

The first line contains the only integer n ($1 \le n \le 100$) — the number of forbidden substrings in the collection. Next n lines contain these substrings. The next line contains string w. All those $n + 1$ lines are non-empty strings consisting of uppercase and lowercase Latin letters whose length does not exceed 100. The last line contains a lowercase letter $letter$.

Output

Output the only line — Petya's resulting string with the maximum number of letters $letter$. If there are several answers then output the one that comes first lexicographically.

The lexicographical comparison is performed by the standard < operator in modern programming languages. The line a is lexicographically smaller than the line b, if a is a prefix of b, or there exists such an i ($1 \le i \le |a|$), that $a_i < b_i$, and for any j ($1 \le j < i$) $a_j = b_j$. $|a|$ stands for the length of string a.

Examples

input

```
3
bers
ucky
elu
PetrLoveLuckyNumbers
t
```

output

```
PetrLovtTttttNumtttt
```

input

```
4
hello
party
abefglghjdhfgj
IVan
petrsmatchwin
a
```

output

```
petrsmatchwin
```

input

```
2
aCa
cba
abAcaba
c
```

output

```
abCacba
```

D. Volleyball

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves volleyball very much. One day he was running late for a volleyball match. Petya hasn't bought his own car yet, that's why he had to take a taxi. The city has n junctions, some of which are connected by two-way roads. The length of each road is defined by some positive integer number of meters; the roads can have different lengths.

Initially each junction has exactly one taxi standing there. The taxi driver from the i-th junction agrees to drive Petya (perhaps through several intermediate junctions) to some other junction if the travel distance is not more than t_i meters. Also, the cost of the ride doesn't depend on the distance and is equal to c_i bourles. Taxis can't stop in the middle of a road. **Each taxi can be used no more than once. Petya can catch taxi only in the junction, where it stands initially.**

At the moment Petya is located on the junction x and the volleyball stadium is on the junction y. Determine the minimum amount of money Petya will need to drive to the stadium.

Input

The first line contains two integers n and m $(1 \le n \le 1000, 0 \le m \le 1000)$. They are the number of junctions and roads in the city correspondingly. The junctions are numbered from 1 to n, inclusive. The next line contains two integers x and y ($1 \le x, y \le n$). They are the numbers of the initial and final junctions correspondingly. Next m lines contain the roads' description. Each road is described by a group of three integers u_i, v_i, w_i ($1 \le u_i, v_i \le n, 1 \le w_i \le 10^9$) — they are the numbers of the junctions connected by the road and the length of the road, correspondingly. The next n lines contain n pairs of integers t_i and c_i ($1 \le t_i, c_i \le 10^9$), which describe the taxi driver that waits at the i-th junction — the maximum distance he can drive and the drive's cost. The road can't connect the junction with itself, but between a pair of junctions there can be more than one road. All consecutive numbers in each line are separated by exactly one space character.

Output

If taxis can't drive Petya to the destination point, print "-1" (without the quotes). Otherwise, print the drive's minimum cost.

Please do not use the %lld specificator to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the %I64d specificator.

Examples

input

```
4 4
1 3
1 2 3
1 4 1
2 4 1
2 3 5
2 7
7 2
1 2
7 7
```

output

```
9
```

Note

An optimal way — ride from the junction 1 to 2 (via junction 4), then from 2 to 3. It costs 7+2=9 bourles.

E. Horse Races

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya likes horse racing very much. Horses numbered from l to r take part in the races. Petya wants to evaluate the probability of victory; for some reason, to do that he needs to know the amount of nearly lucky horses' numbers. A *nearly lucky* number is an integer number that has at least two lucky digits the distance between which does not exceed k. Petya learned from some of his mates from Lviv that lucky digits are digits 4 and 7. The distance between the digits is the absolute difference between their positions in the number of a horse. For example, if $k = 2$, then numbers 412395497, 404, 4070400000070004007 are nearly lucky and numbers 4, 4123954997, 4007000040070004007 are not.

Petya prepared t intervals $[l_i, r_i]$ and invented number k, common for all of them. Your task is to find how many nearly happy numbers there are in each of these segments. Since the answers can be quite large, output them modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains two integers t and k $(1 \le t, k \le 1000)$ — the number of segments and the distance between the numbers correspondingly. Next t lines contain pairs of integers l_i and r_i $(1 \le l \le r \le 10^{1000})$. All numbers are given without the leading zeroes. Numbers in each line are separated by exactly one space character.

Output

Output t lines. In each line print one integer — the answer for the corresponding segment modulo 1000000007 $(10^9 + 7)$.

Examples

input

```
1 2
1 100
```

output

```
4
```

input

```
1 2
70 77
```

output

```
2
```

input

```
2 1
1 20
80 100
```

output

```
0
0
```

Note

In the first sample, the four nearly lucky numbers are 44, 47, 74, 77.

In the second sample, only 74 and 77 are in the given segment.