Codeforces Beta Round #84 (Div. 2 Only)

A. Nearly Lucky Number

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

*Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Unfortunately, not all numbers are lucky. Petya calls a number *nearly lucky* if the number of lucky digits in it is a lucky number. He wonders whether number n is a nearly lucky number.

Input
The only line contains an integer n ($1 \le n \le 10^{18}$).

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

Output
Print on the single line "YES" if n is a nearly lucky number. Otherwise, print "NO" (without the quotes).

Examples

input
40047
output
NO

input
7747774
output
YES

input
1000000000000000000
output
NO

Note
In the first sample there are 3 lucky digits (first one and last two), so the answer is "NO".

In the second sample there are 7 lucky digits, 7 is lucky number, so the answer is "YES".

In the third sample there are no lucky digits, so the answer is "NO".

B. Lucky String

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

*Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Petya recently learned to determine whether a string of lowercase Latin letters is lucky. For each individual letter all its positions in the string are written out in the increasing order. This results in 26 lists of numbers; some of them can be empty. A string is considered lucky if and only if in each list the absolute difference of any two **adjacent** numbers is a lucky number.

For example, let's consider string "zbcdzefdzc". The lists of positions of equal letters are:

- b: 2
- c: 3, 10
- d: 4, 8
- e: 6
- f: 7
- z: 1, 5, 9
- Lists of positions of letters a, g, h, ..., y are empty.

This string is lucky as all differences are lucky numbers. For letters z: 5 - 1 = 4, 9 - 5 = 4, for letters c: 10 - 3 = 7, for letters d: 8 - 4 = 4.

Note that if some letter occurs only once in a string, it doesn't influence the string's luckiness after building the lists of positions of equal letters. The string where all the letters are distinct is considered lucky.

Find the lexicographically minimal lucky string whose length equals n.

Input
The single line contains a positive integer n ($1 \le n \le 10^5$) — the length of the sought string.

Output
Print on the single line the lexicographically minimal lucky string whose length equals n.

Examples

input
5
output
abcda

input
3
output
abc

Note
The lexical comparison of strings is performed by the < operator in modern programming languages. String a is lexicographically less than string b if exists such i ($1 \le i \le n$), that $a_i < b_i$, and for any j ($1 \le j < i$) $a_j = b_j$.

C. Lucky Sum of Digits

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

*Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Petya wonders eagerly what minimum lucky number has the sum of digits equal to n. Help him cope with the task.

Input
The single line contains an integer n ($1 \le n \le 10^6$) — the sum of digits of the required lucky number.

Output
Print on the single line the result — the minimum lucky number, whose sum of digits equals n. If such number does not exist, print -1.

Examples

input
11
output
47

input
10
output
-1

D. Lucky Probability

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

*Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Petya and his friend Vasya play an interesting game. Petya randomly chooses an integer p from the interval $[p_l, p_r]$ and Vasya chooses an integer v from the interval $[v_l, v_r]$ (also randomly). Both players choose their integers equiprobably. Find the probability that the interval $[min(v, p), max(v, p)]$ contains exactly k lucky numbers.

Input
The single line contains five integers p_l, p_r, v_l, v_r and k ($1 \le p_l \le p_r \le 10^9$, $1 \le v_l \le v_r \le 10^9$, $1 \le k \le 1000$).

Output
On the single line print the result with an absolute error of no more than 10^{-9}.

Examples

input
1 10 1 10 2
output
0.320000000000

input
5 6 8 10 1
output
1.000000000000

Note
Consider that $[a, b]$ denotes an interval of integers; this interval **includes** the boundaries. That is, $[a, b] \stackrel{def}{=} \{x \in \mathbb{R} : a \leqslant x \leqslant b\}$

In first case there are 32 suitable pairs:
$(1, 7), (1, 8), (1, 9), (1, 10), (2, 7), (2, 8), (2, 9), (2, 10), (3, 7), (3, 8), (3, 9), (3, 10), (4, 7), (4, 8), (4, 9), (4, 10), (7, 1), (7, 2), (7, 3), (7, 4), (8, 1), (8,$. Total number of possible pairs is $10 \cdot 10 = 100$, so answer is $32 / 100$.

In second case Petya always get number less than Vasya and the only lucky 7 is between this numbers, so there will be always 1 lucky number.

E. Lucky Tree

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

*Petya loves lucky numbers. We all know that lucky numbers are the positive integers whose decimal representations contain only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

One day Petya encountered a tree with n vertexes. Besides, the tree was weighted, i. e. each edge of the tree has weight (a positive integer). An edge is lucky if its weight is a lucky number. Note that a *tree with n vertexes* is an undirected connected graph that has exactly $n - 1$ edges.

Petya wondered how many vertex triples (i, j, k) exists that on the way from i to j, as well as on the way from i to k there must be at least one lucky edge (all three vertexes are pairwise distinct). The order of numbers in the triple matters, that is, the triple $(1, 2, 3)$ is not equal to the triple $(2, 1, 3)$ and is not equal to the triple $(1, 3, 2)$.

Find how many such triples of vertexes exist.

Input

The first line contains the single integer n ($1 \le n \le 10^5$) — the number of tree vertexes. Next $n - 1$ lines contain three integers each: u_i v_i w_i ($1 \le u_i, v_i \le n, 1 \le w_i \le 10^9$) — the pair of vertexes connected by the edge and the edge's weight.

Output

On the single line print the single number — the answer.

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

Examples

input
4 1 2 4 3 1 2 1 4 7
output
16

input
4 1 2 4 1 3 47 1 4 7447
output
24

Note

The 16 triples of vertexes from the first sample are:
$(1, 2, 4), (1, 4, 2), (2, 1, 3), (2, 1, 4), (2, 3, 1), (2, 3, 4), (2, 4, 1), (2, 4, 3), (3, 2, 4), (3, 4, 2), (4, 1, 2), (4, 1, 3), (4, 2, 1), (4, 2, 3), (4, 3, 1), (4, 3, 2)$.

In the second sample all the triples should be counted: $4 \cdot 3 \cdot 2 = 24$.