Codeforces Beta Round #91 (Div. 1 Only)

A. Lucky Sum

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 lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.*

Let $next(x)$ be the minimum lucky number which is larger than or equals x. Petya is interested what is the value of the expression $next(l) + next(l + 1) + ... + next(r - 1) + next(r)$. Help him solve this problem.

Input

The single line contains two integers l and r ($1 \le l \le r \le 10^9$) — the left and right interval limits.

Output

In the single line print the only number — the sum $next(l) + next(l + 1) + ... + next(r - 1) + next(r)$.

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.

Sample test(s)

input
2 7
output
33

input
7 7
output
7

Note

In the first sample: $next(2) + next(3) + next(4) + next(5) + next(6) + next(7) = 4 + 4 + 4 + 7 + 7 + 7 = 33$

In the second sample: $next(7) = 7$

B. Lucky Transformation

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 lucky numbers are positive integers whose decimal representation contains only the lucky digits ***4*** *and* ***7****. For example, numbers* ***47****,* ***744****,* ***4*** *are lucky and* ***5****,* ***17****,* ***467*** *are not.*

Petya has a number consisting of n digits without leading zeroes. He represented it as an array of digits without leading zeroes. Let's call it d. The numeration starts with 1, starting from the most significant digit. Petya wants to perform the following *operation* k times: find the minimum x $(1 \le x < n)$ such that $d_x = 4$ and $d_{x+1} = 7$, if x is odd, then to assign $d_x = d_{x+1} = 4$, otherwise to assign $d_x = d_{x+1} = 7$. Note that if no x was found, then the operation counts as completed and the array doesn't change at all.

You are given the initial number as an array of digits and the number k. Help Petya find the result of completing k operations.

Input

The first line contains two integers n and k $(1 \le n \le 10^5, 0 \le k \le 10^9)$ — the number of digits in the number and the number of completed operations. The second line contains n digits without spaces representing the array of digits d, starting with d_1. It is guaranteed that the first digit of the number does not equal zero.

Output

In the single line print the result without spaces — the number after the k operations are fulfilled.

Sample test(s)

input
7 4 4727447
output
4427477

input
4 2 4478
output
4478

Note

In the first sample the number changes in the following sequence: $4727447 \to 4427447 \to 4427477 \to 4427447 \to 4427477$.

In the second sample: $4478 \to 4778 \to 4478$.

C. Lucky Permutation

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 lucky numbers are positive integers whose decimal representation contains only the lucky digits ***4*** *and* ***7****. For example, numbers* ***47****,* ***744****,* ***4*** *are lucky and* ***5****,* ***17****,* ***467*** *are not.*

One day Petya dreamt of a lexicographically k-th permutation of integers from 1 to n. Determine how many lucky numbers in the permutation are located on the positions whose indexes are also lucky numbers.

Input

The first line contains two integers n and k $(1 \le n, k \le 10^9)$ — the number of elements in the permutation and the lexicographical number of the permutation.

Output

If the k-th permutation of numbers from 1 to n does not exist, print the single number "-1" (without the quotes). Otherwise, print the answer to the problem: the number of such indexes i, that i and a_i are both lucky numbers.

Sample test(s)

input
7 4
output
1

input
4 7
output
1

Note

A permutation is an ordered set of n elements, where each integer from 1 to n occurs exactly once. The element of permutation in position with index i is denoted as a_i $(1 \le i \le n)$. Permutation a is lexicographically smaller that permutation b if there is such a i $(1 \le i \le n)$, that $a_i < b_i$, and for any j $(1 \le j < i)$ $a_j = b_j$. Let's make a list of all possible permutations of n elements and sort it in the order of lexicographical increasing. Then the lexicographically k-th permutation is the k-th element of this list of permutations.

In the first sample the permutation looks like that:

```
1 2 3 4 6 7 5
```

The only suitable position is 4.

In the second sample the permutation looks like that:

```
2 1 3 4
```

The only suitable position is 4.

D. Lucky Segments

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits ***4*** *and* ***7****. For example, numbers* ***47****,* ***744****,* ***4*** *are lucky and* ***5****,* ***17****,* ***467*** *are not.*

Petya has n number segments $[l_1; r_1]$, $[l_2; r_2]$, ..., $[l_n; r_n]$. During one move Petya can take any segment (let it be segment number i) and replace it with segment $[l_i + 1; r_i + 1]$ or $[l_i - 1; r_i - 1]$. In other words, during one move Petya can shift any segment to the left or to the right by a unit distance. Petya calls a number *full* if it belongs to each segment. That is, number x is full if for any i $(1 \le i \le n)$ the condition $l_i \le x \le r_i$ is fulfilled.

Petya makes no more than k moves. After that he counts the quantity of full lucky numbers. Find the maximal quantity that he can get.

Input

The first line contains two integers n and k $(1 \le n \le 10^5, 1 \le k \le 10^{18})$ — the number of segments and the maximum number of moves. Next n lines contain pairs of integers l_i and r_i $(1 \le l_i \le r_i \le 10^{18})$.

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

Output

Print on the single line the single number — the answer to the problem.

Sample test(s)

input
4 7 1 4 6 9 4 7 3 5

output
1

input
2 7 40 45 47 74

output
2

Note

In the first sample Petya shifts the second segment by two units to the left (it turns into $[4; 7]$), after that number 4 becomes full.

In the second sample Petya shifts the first segment by two units to the right (it turns into $[42; 47]$), and shifts the second segment by three units to the left (it turns into $[44; 71]$), after that numbers 44 and 47 become full.

E. Lucky Array

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits ***4*** *and* ***7****. For example, numbers* ***47****,* ***744****,* ***4*** *are lucky and* ***5****,* ***17****,* ***467*** *are not.*

Petya has an array consisting of n numbers. He wants to perform m operations of two types:

- *add* l r d — add an integer d to all elements whose indexes belong to the interval from l to r, inclusive $(1 \le l \le r \le n, 1 \le d \le 10^4)$;
- *count* l r — find and print on the screen how many lucky numbers there are among elements with indexes that belong to the interval from l to r inclusive $(1 \le l \le r \le n)$. Each lucky number should be counted as many times as it appears in the interval.

Petya has a list of all operations. The operations are such that after all additions the array won't have numbers that would exceed 10^4. Help Petya write a program that would perform these operations.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of numbers in the array and the number of operations correspondingly. The second line contains n positive integers, none of which exceeds 10^4 — those are the array numbers. Next m lines contain operations, one per line. They correspond to the description given in the statement.

It is guaranteed that after all operations are fulfilled each number in the array will not exceed 10^4.

Output

For each operation of the second type print the single number on the single line — the number of lucky numbers in the corresponding interval.

Sample test(s)

input

```
3 6
2 3 4
count 1 3
count 1 2
add 1 3 2
count 1 3
add 2 3 3
count 1 3
```

output

```
1
0
1
1
```

input

```
4 5
4 4 4 4
count 1 4
add 1 4 3
count 1 4
add 2 3 40
count 1 4
```

output

```
4
4
4
```

Note

In the first sample after the first addition the array will look in the following manner:

```
4 5 6
```

After the second addition:

```
4 8 9
```

The second sample after the first addition:

```
7 7 7 7
```

After the second addition:

```
7 47 47 7
```