problem	zvrk	xl	deva	vid

source file	`zvrk.pas`
	`zvrk.c`
	`zvrk.cpp`
		`xl.pas`
		`xl.c`
		`xl.cpp`
			`deva.pas`
			`deva.c`
			`deva.cpp`
				`vid.pas`
				`vid.c`
				`vid.cpp`

input data	stdin	stdin	library	stdin
output data	stdout	stdout	library	stdout
time limit (Athlon 64 3000+)	1 sec	1 sec	5 sec	2 sec
memory limit (heap)	32 MB	32 MB	64 MB	64 MB
memory limit (stack)	8 MB	8 MB	16 MB	16 MB
points	**50**	**60**	**90**	**100**
	300			

By **removing all digits left of the rightmost digit one** in the **binary representation** of some integer, we get what is called the "**zvrk**" of that number.

For example, the zvrk of 6 i.e. $(110)_2$ is 2 i.e. $(10)_2$, and the zvrk of 40 i.e. $(101000)_2$ is 8 i.e. $(1000)_2$.

Write a program that will calculate **the sum of the zvrks** of all numbers **between** two given numbers A and B (**inclusive**).

input data

First and only line of input contains two integers A and B, $1 \leq A \leq B \leq 10^{15}$.

output data

First and only line of output should contain the sum from the problem statement.

Note: the result will fit into the 64-bit signed integer type (int64 in Pascal, long long in C/C++).

examples

input	input	input
176 177	5 9	25 28
output	**output**	**output**
17	13	8

A square consists of cells organized in N rows and N columns. Each cell is either **empty** or contains a **single Roman numeral** (I, V, X, L, C, or D, representing numbers 1, 5, 10, 50, 100 and 500 respectively).

Look at the picture from third test example:

I	I	X	V		L	X
X	L		I	X	V	I
	I	V	I	X		X
L	I	X		V	I	X
X		X	I	X	I	
L	I	V	L		X	X
V	I		X	I	X	L

We start at the **central cell** of the square (N will be odd and central cell will be empty in all test cases) and try to find the **longest sequence** of Roman numerals representing consecutive integers starting from 1 and separated by **one empty cell**. A sequence is obtained by "walking" through the square so that each step consists of moving to an adjacent cell left, right, up or down from the current cell. Each Roman numeral (including the last one) in a sequence must end with one (and only one) empty cell in a sequence. The output of a program should be (a decimal representation of) the largest number in the sequence.

Starting with the square above we can walk up-down-up-up-left-right-down-up-left to obtain the first tree numbers in a sequence IeIIeIIIe (where e denotes empty cell).

To obtain Roman representation of a number written in a decimal notation one may proceed as follows: Convert each decimal digit separately to its Roman equivalent taking its positional value into account, in descending order by their values. E.g. 726 = 700+20+6 → DCC XX VI = DCCXXVI. There are some special cases when letters representing Roman numeral of lower values are written before letters representing Roman numeral of higher values. Special cases here are numbers 4, 9, 40, 90 and 400 whose Roman numerals are obtained by writing lower numeral before higher: IV, IX, XL, XC and CD respectively. E.g. 499 = 400+90+9 = CD XC IX = CDXCIX (and not ID, as it might appear to be).

Write a program that will find the longest sequence of Roman numerals representing consecutive positive integer numbers starting from 1 and separated by one empty cell (as described above) and write (a decimal representation of) the largest number in the sequence.

input data

First line of input contains an odd integer N, $1 \leq N \leq 99$.

Each of the next N lines contains N characters depicting the corresponding row of a square using characters 'I', 'V', 'X', 'L', 'C', 'D' and '.' (dot, denoting empty cell).

output data

First and only line of output should contain the decimal representation of the last number in the longest sequence of Roman numerals as described above.

examples

input

```
3
I.I
I.V
I..
```

output

```
6
```

input

```
5
.IV.I
II.II
XV.VI
..I..
...II
```

output

```
11
```

input

```
7
IIXV.LX
XL.IXVI
.IVIX.X
LIX.VIX
X.XIXI.
LIVL.XX
VI.XIXL
```

output

```
51
```

A desert contains N cities, $2 \leq N \leq 1000000$ (one million). Some of them are directly connected by roads, and each city is connected directly with **at most 5 other cities**.

Abdul and Yasser are given a list of K cities, $2 \leq K \leq 8$, which they must visit riding on a camel. The cities are marked with $\mathbf{a_1}$ through $\mathbf{a_K}$, and the camel they are riding can traverse **at most L roads per day**, $1 \leq L \leq 10$.

They **have to start** their journey in city $\mathbf{a_1}$ and then visit and spend the night **in each of the other** K-1 cities **in exactly K days**. They may visit them **in any order**, but **return** to city $\mathbf{a_1}$ at the end of the trip. They may not visit any of the cities $\mathbf{a_2}$ through $\mathbf{a_K}$ more than once.

However, Abdul and Yasser don't have a map of the desert so they cannot know which cities are connected by roads, and which aren't. But, by using an SMS service of the local mobile telephony provider, they can find out **all cities which are directly connected** to some city. Seeing as our protagonists are on the run, their funding is limited and so they can send **at most 5000** such SMS messages.

Write a program which will help them find some route adhering to the given constraints.

library

To solve the problem you will use a library (`devalib`) containing four functions.

`Init` - needs to be called **exactly once** and **before** using any of the other library functions. This function returns the numbers N and L.

```
procedure Init(var n, l : longint);
void Init(int *n, int *l);
```

`Gradovi` - returns the number and list of cities Abdul and Yasser need to visit, $\mathbf{a_1}$ through $\mathbf{a_k}$, respectively.

```
procedure Cities(var k : longint; var listc: array of longint);
void Cities(int *k, int listc[]);
```

`Susjedi` - simulates the sending of one SMS message. Returns the number and list of cities connected by road to the given city *city*. You may call this function **at most 5000 times**.

```
procedure Neighbours(city : longint; var howmany : longint; var neighbours:
array of longint);
void Neighbours(int city, int *howmany, int neighbours[]);
```

`Rjesenje` - needs to be called **at the end** of your program. Its arguments are the cities Abdul and Yasser visited, in order. The first and last city in the list needs to be city $\mathbf{a_1}$. This function will also **regularly end** your program's execution.

```
procedure Solution(tourlength : longint; var tour : array of longint);
void Solution(int tourlength, int tour[]);
```

N points are given in a plane. Any point has **integer** coordinates and **no two points** have the same x coordinates or the same y coordinates.

A pair of points A and B uniquely define a rectangle R(A,B) whose sides are parallel to coordinate axes such that the points A and B are endpoints of one of its diagonals. A pair of two given points A and B are **very visible** if there are **no other given points within a rectangle** R(A,B). A pair of points consists of two different points and in this problem pairs (A,B) and (B,A) are regarded to be the same pair and are counted as one pair.

At the beginning a coordinate plane has no given points. Your program should read coordinates of the given points from a file and after reading coordinates of a point, it should write number of pairs of currently very visible given points.

input data

First line of input contains an integer N, $2 \leq N \leq 5000$, the number of given points.

Each of the following N lines contains two integers X, Y, $0 \leq X,Y \leq 1000000$ (one million), coordinates of the given points in the order they should be added to the plane.

output data

An output should contain N lines, k^{th} line should contain a number of pairs of very visible given points after taking k^{th} given point into account.

examples

input

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

output

```
0
1
3
```

input

```
4
2 3
3 4
1 2
4 1
```

output

```
0
1
2
5
```

input

```
7
1 5
2 7
3 8
5 1
6 2
7 3
4 4
```

output

```
0
1
2
5
9
13
10
```