Game of Primes

In a global Mathematics contest, the contestants are told to invent some special numbers which can be built by adding the squares of its digits. Doing this repeatedly, the numbers will end up to either 1 or 4.
If a positive integer ends with 1, then it is called the Number of Game.
An example from above is:
13 = 1^2 + 3^2 = 1+9 = 10 (Step:1)
10 = 1^2 + 0^2 = 1+0 = 1 (Step:2), iteration ends in Step 2 since number ends with 1

A number is considered special if the sum of squares of its digits (repeatedly) is 1 and the number is a prime number.
Write a program to find out all the special numbers within any given range (inclusive of both the limits).

Input Format:
The first line consists of a single integer that denotes the lower limit of the range.
The first line consists of a single integer that denotes the upper limit of the range.

Output Format:
Print all the special numbers in ascending order (one in each line).
Print "No special number present in the given range" (without the quotes).

Sample Input 1:
1
15

Sample Output 1:
7
13

Explanation:
Prime numbers between 1 and 15
2,3,5,7,13

Number	Sum of Squares of digits
2	4
3	4
5	4
7	1
13	1

For example,
Sum of Squares of digits of - 3
3

3*3 = 9
9*9 = 81
8*8 + 1*1 = 65
6*6 + 5*5 = 61
6*6 + 1*1 = 37
3*3 + 7*7 = 58
5*5 + 8*8 = 89
8*8 + 9*9 = 145
1*1+4*4+5*5 = 42
4*4 + 2*2 = 20
2*2 +0*0 = 4
Sum of squares of digits of 3 = 4

The special numbers between 1 and 15 are 7 and 13

Sample Input 2:
1
5

Sample Output 2:
No special number present in the given range

SNo	Name	Input	Output
1	ST2	1 5	No special number present in the given range
2	ST1	1 15	7 13
3	T4	10 35	13 19 23 31
4	T7	1 1	No special number present in the given range
5	T5	20 200	23 31 79 97 103 109 139 167 193

SNo	Name	Input	Output
6	T3	1 30	7 13 19 23
7	T6	7 19	7 13 19
8	T8	200 220	No special number present in the given range

```
import java.util.*;
class Main  {
        public static void main(String args[]) {
                Scanner sc=new Scanner(System.in);
                int n=sc.nextInt();
                int m=sc.nextInt();
                int count=0;
                for(int i=n;i<=m;i++)
                {
               if(check(i) && i!=1)
                           {
                                   int k=i;
                                   if(recur(k)==1)
{System.out.println(i);count++;}
                                   else continue;
                           }
                }
                if(count==0) System.out.println("No special number present in
the given range");
        }
        static boolean check(int n)
        {
                for(int i=2;i<=n/2+1;i++){
                      if(n%i==0) return false;
              }
                   return true;
        }
        static int recur(int n)
        {
                int sum=0;
                if(n==10) return 1;
                int p=n;
                        while(p>0)
                        {
                        int r=p%10;
                        sum+=Math.pow(r,2);
```

```
                        p/=10;
                        }
                //System.out.println(sum);
                        if(sum==1) return 1;
                        else if(sum==4) return 0;
                        return recur(sum);
        }
}
```

BASE 6

Given a sequence of distinct numbers a1, a2, ….. an, an inversion occurs if there are indices i<j such that ai > aj .

For example, in the sequence 2 1 4 3 there are 2 inversions (2 1) and (4 3).

The input will be a main sequence of N positive integers. From this sequence, a Derived Sequence will be obtained using the following rule. The output is the number of inversions in the derived sequence.

Rule for forming derived sequence
An integer may be represented in base 6 notation. In base 6, 10305 is 1x6^4 + 3x6^2 + 5 =1409. Note that none of the digits in that representation will be more than 5.
The sum of digits in a base 6 representation is the sum of all the digits at the various positions in the representation. Thus for the number 1409, the representation is 10305, and the sum of digits is 1+0+3+0+5=9. The sum of digits may be done in the decimal system, and does not need to be in base 6
The derived sequence is the sum of the digits when the corresponding integer is represented in the base 6 form number will be expressed in base 6, and the derived sequence is the sum of the digits of the number in the base 6 representation.

Input Format:
The first line of the input will have a single integer, which will give N.
The next line will consist of a comma separated string of N integers, which is the main sequence

Output Format:
The number of inversions in the derived sequence formed from the main sequence.

Constraints:
N <= 50
Integers in sequence <= 107

Sample Input 1:
5
55, 53, 88, 27, 33

Sample Output 1:
2

Sample Input 2:
8
120,21,47,64,72,35,18,98

Sample Output 2:
11

Explanation for example 1:
The number of integers is 5, as specified in the first line. The given sequence is 55, 53, 88, 27, 33. The base6 representation is 131, 125, 224, 43, 53 The derived sequence is 5,8,8,7,8 (corresponding to the sum of digits). The number of inversions in this is 2, namely (8, 7), (8, 7)
Explanation for example 2:
The base 6 representation of this is 320,33,115,144,200,55,30,242, and the derived sequence (sum of digits) is 5,6,7,9,2,10,3,8. The number of inversions is 11 (5,2), (5,3),(6,2) (6,3), (7,2), (7,3) (9,2),(9,3) (9,8),(10,3), (10,8)

SNo	Name	Input	Output
1	TC1	5 55, 53, 88, 27, 33	2
2	TC2	8 120,21,47,64,72,35,18,98	11
3	TC6	7 56,76,89,23,45,33,23	7
4	TC7	5 100,90,87,101,54	6
5	TC4	2 5,6	1
6	TC3	4 22,55,66,88	2
7	TC5	1 5	0

```
def count(seq):
    k=0;
    for i in range(len(seq)):
        for j in range(i+1,len(seq)):
            if(seq[i]>seq[j]):
                k+=1
    return k
def sumDigits(num):
    s=''
    while num>0:
        s=str(num%6)+s
        num//=6
    return sum(map(int,s))
n=int(input())
mseq=list(map(int,input().split(',')))
dseq=[sumDigits(num) for num in mseq]
result=count(dseq)
print(result)
```

Saving For Rainy Day

By nature, an average Indian believes in saving money. Some reports suggest that an average Indian manages to save approximately 30+% of his salary. Dhaniram is one such hard working fellow. With a view of future expenses, Dhaniram resolves to save a certain amount in order to meet his cash flow demands in the future.
Consider the following example.
Dhaniram wants to buy a TV. He needs to pay Rs 2000/- per month for 12 installments to own the TV. If let's say he gets 4% interest per annum on his savings bank account, then Dhaniram will need to deposit a certain amount in the bank today, such that he is able to withdraw Rs 2000/- per month for the next 12 months without requiring any additional deposits throughout.

Your task is to find out how much Dhaniram should deposit today so that he gets assured cash flows for a fixed period in the future, given the rate of interest at which his money will grow during this period.

Input Format:
First line contains desired cash flow **M**
Second line contains period in months denoted by **T**
Third line contains rate per annum **R** expressed in percentage at which deposited amount will grow

Output Format:
Print total amount of money to be deposited now rounded off to the nearest integer

Constraints:
M > 0
T > 0
R >= 0
Calculation should be done upto 11-digit precision

Sample Input 1:

500
3
12
Sample Output 1:
1470

Sample Input 2:

6000
3
5.9
Sample Output 2:

17824

SNo	Name	Input	Output
1	TC02	6000 3 5.9	17824
2	TC04	10000 12 7.5	112280
3	TC05	100 1 1	100
4	TC01	500 3 12	1470
5	TC03	500 2 0	1000

```
import java.util.*;
class Main  {
        public static void main(String args[]) {
                Scanner s=new Scanner(System.in);
                float m=s.nextFloat();
                int t=s.nextInt();
                float r=s.nextFloat();
                float x=m*t*1200/(1200+r*(t-1));
                System.out.println((int)x);
        }
}
```

Reverse Gear

A futuristic company is building an autonomous car. The scientists at the company are training the car to perform Reverse parking. To park, the car needs to be able to move in backward as well as forward direction. The car is programmed to move backwards B meters and forwards again, say F meters, in a straight line. The car does this repeatedly until it is able to park or collides with other objects. The car covers 1 meter in T units of time. There is a wall after distance D from car's initial position in the backward direction.

The car is currently not without defects and hence often hits the wall. The scientists are devising a strategy to prevent this from happening. Your task is to help the scientists by

providing them with exact information on amount of time available before the car hits the wall.

Input Format:

First line contains total number of test cases, denoted by N

Next N lines, contain a tuple containing 4 values delimited by space

F B T D, where

1. **F** denotes forward displacement in meters
2. **B** denotes backward displacement in meters
3. **T** denotes time taken to cover 1 meter
4. **D** denotes distance from Car's starting position and the wall in backward direction

Output Format:

For each test case print time taken by the Car to hit the wall

Constraints:

First move will always be in backward direction

1 <= N <= 100

backward displacement > forward displacement i.e. (B > F)

forward displacement (F) > 0

backward displacement (B) > 0

time (T) > 0

distance (D) > 0

All input values must be positive integers only

Sample Input:

6 9 3 18

Sample Output:

162

SNo	Name	Input	Output
1	TC05	1 2 1 1	1

SNo	Name	Input	Output
2	TC02	3 7 5 20	220
3	TC01	6 9 3 18	162
4	TC04	10 15 2 25	130
5	TC03	5 8 3 15	135

```
f,b,t,dt=list(map(int,input().split()))
temp=b
c,d=0,0
if b>f:
    while temp<dt:
        temp-=f
        c+=1
        if temp<dt:
            temp+=b
            d+=1
    if temp==dt:
        res=(c*f+d*b)*t+b*t
    elif b==f and b>dt:
        res=d*t
    else:
        fk=(c*f+d*b)*t+b*t
        temp=temp-dt
        res=fk-(temp*t)
print(res)
```

TRACE THE RATS

Given a square maze (A) of dimension N, every entry (Aij) in the maze is either an open cell 'O' or a wall 'X'. A rat can travel to its adjacent locations (left, right, top and bottom), but to reach a cell, it must be open. Given the locations of rats, can you find out whether all the rats can reach others or not?

Input Format:
Input will consist of three parts, viz.
1. Size of the maze (N)
2. The maze itself (A = N * N)
3.Number of rats (R)
4. Location of R rats (Xi, Yi)

Note:
(Xi,Yi) will represent the location of the ith rat.
Locations are 1-index based.

Output Format:
Print "Yes" if the rats can reach each other, else print "No".

Constraints:
1<=N<=350
Aij = {'O','X'}
1<=R<=N*N
1<=Xi<=N
1<=Yi<=N

Sample Input 1:
3
OOX
OXO
OOX
4
1 1
1 2
2 1
3 2
Sample Output 1:
Yes

Sample Input 2:
3
OOX
OXO
OOX
4
1 1
1 2
2 1
2 3
Sample Ouput 2:
No

SNo	Name	Input	Output
1	TC3	3 OOX OXO OOX 4 1 1 2 2	NO

SNo	Name	Input	Output
		2 2 2 3	
2	TC4	3 OXX OXO XOX 4 1 1 2 2 2 2 2 3	NO
3	TC2	3 OOX OXO OOX 4 1 1 1 2 2 1 2 3	NO
4	TC5	3 OOX OOO OOX 4 1 1 1 1 1 2 3 2	YES
5	TC1	3 OOX OXO OOX 4 1 1 1 2 2 1 3 2	YES

```
import java.util.*;
class Main  {
        static boolean flag=false;
        public static void help(char a[][],int x,int y, int n)
        {
```

```
		if(flag){
			return ;
		}
		if(x<0 || y<0 || x>=n || y>=n || a[x][y]=='X')
		{
			return;
		}
		if(a[x][y] == 'O')
		{
			flag= true;
			return;
		}
	}


	public static void main(String args[]) {


		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		char a[][]=new char[n][n];
		for(int i=0;i<n;i++)
		{
			String s=sc.next();
			a[i]=s.toCharArray();
		}

		int m=sc.nextInt();
		int x=0,y=0;
		for(int i=0;i<m;i++)
		{
			x=sc.nextInt()-1;
			y=sc.nextInt()-1;
			flag=false;
			help(a,x+1,y,n);
			help(a,x-1,y,n);
			help(a,x,y+1,n);
			help(a,x,y-1,n);
			if(!flag)
			{
				System.out.print("NO");
				return;
			}
		}
		System.out.println("YES");

	}
}
```

Counting the Rational Numbers Problem

Georg Cantor proved that the rational numbers form a countably infinite set - that is, we can define a one to one correspondence between the rational numbers and the set of positive integers. One way of showing that the rational numbers are countable is by forming a two-dimensional array as follows:

Such an array would cover all rational numbers. The blank spaces shown as "-" indicate fractions that already appear in their lowest form earlier in the array. One can now count the rationals diagonal after diagonal as follows:

1/1, 2/1, 1/2, 1/3, (2/2 skipped), 3/1, 4/1, 3/2, 2/3, 1/4, 1/5, (2/4, 3/3, 4/2 skipped), 5/1...

Write a program for perform such a counting of positive rational numbers and output the sequence number N corresponding to the input rational number Q. Note that if the input is not in its lowest form, it needs to be reduced to its lowest form before searching.

Input

A line containing a rational number in the form n/m, where n and m are positive integers

Output

The count of that number using the above approach

Constraints

$1 \leq n, m \leq 100$

Example 1

Input
3/1

Output
5

Explanation

Counting the rationals up to 3/1 as per the above procedure and skipping rationals that are not in their lowest forms, 3/1 is seen to be the 5th in the countable list. Hence N = 5.

Example 2

Input
2/3

Output
8

Explanation
The input rational is 2/3. Using the above procedure, the count is 8. Hence the output is 8.

SNo	Name	Input	Output
1	TC5	4/7	38
2	TC2	2/3	8
3	TC3	1/2	3
4	TC1	3/1	5
5	TC6	5/9	60
6	TC4	6/10	19

Partial Answer

```
n=input()
count=0
l=['1/1' , '2/1', '1/2' ,'1/3','3/1','4/1','3/2','2/3','1/4','1/5','5/1']
for i in l:
    if(i==n):
        break
    count=count+1

print(count+1)
```