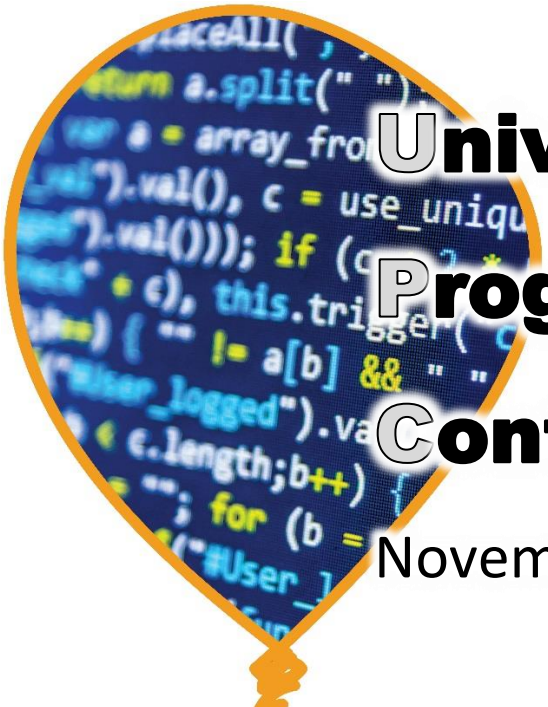
University of Sharjah
College of Science
Department of Computer Science



University Programming Contest

November 11, 2018

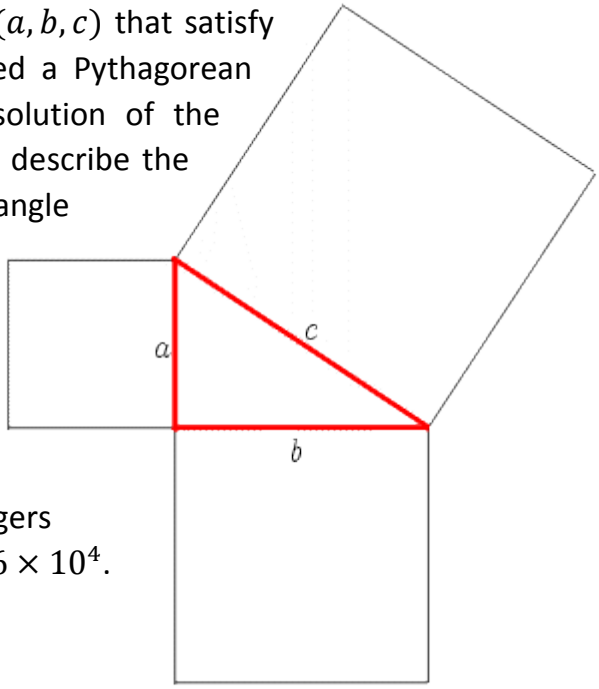
Problem Set

Problem	Balloon
Problem A: Pythagorean Triples	Orange
Problem B: Room Validator	Silver
Problem C: Abacus	Green
Problem D: Unsocial	Red
Problem E: Collatz Conjecture	Purple
Problem F: IComparer Interface	Yellow
Problem G: Football	White
Problem H: Circular Formula	Pink
Problem I: Decryption Order	Black
Problem J: Number Seven	Blue

Do not open before the contest starts

Problem A: Pythagorean Triples

Balloon color: Orange



A triple of three positive integers (a, b, c) that satisfy the equation $a^2 + b^2 = c^2$ is called a Pythagorean triple. Such triple is an integer solution of the Pythagorean Theorem. If the triple describe the sides of a triangle, it will be a right angle triangle.

Given three positive integers find out whether they form a Pythagorean triple or not.

Input format:

The input consists of 3 positive integers a, b, and c such that: $0 < a, b, c \leq 6 \times 10^4$.

Output format:

Output "Yes" if a, b, and c form a Pythagorean triple (in any order); "No" otherwise.

Sample input/output:

Input	Output
4 5 3	Yes
17 55 43	No
8 15 17	Yes
36100 10108 34656	Yes

Problem B: Room Validator

Balloon color: Silver

As agreed in the last CS Department meeting, senior students will be allowed to reserve rooms for their research activities. To streamline this process, we would like to design an online form for reservations. The form should collect the name of the student, email, mobile number and, naturally, the room to be reserved. Data validators should be used for all collected details. WHAT? YOU DON'T KNOW WHAT DATA VALIDATORS ARE?!? A data validator is a set of rules applied to entered data to ensure the correctness and meaningfulness of input. An email validator for instance checks that the text entered is in correct format (someone@domain.com). A mobile number validator checks that the entered text is in the form 05XXXXXXXX (where every X is a single digit).

You are assigned the task of writing the program for room name validator. You need to implement the following rules:

1- Room names consist of two parts: building name and room number separated by a single hyphen (–).
2- Building name starts with capital letter 'M' or 'W' followed by a number in the range 1 to 99 inclusive (i.e. 1 and 99 are valid). No leading zeros.
3- Room number should consist of exactly three digits.
4- No spaces or any other characters are allowed.

Thus, room names such as M3-114, M21-024 and W8-105 are valid while B3-114, M21-24, W08-105, W5 213 and m4-104 are invalid.

Input format:

Each test case is a single line of text T where $1 \leq length(T) \leq 1000$.

Output format:

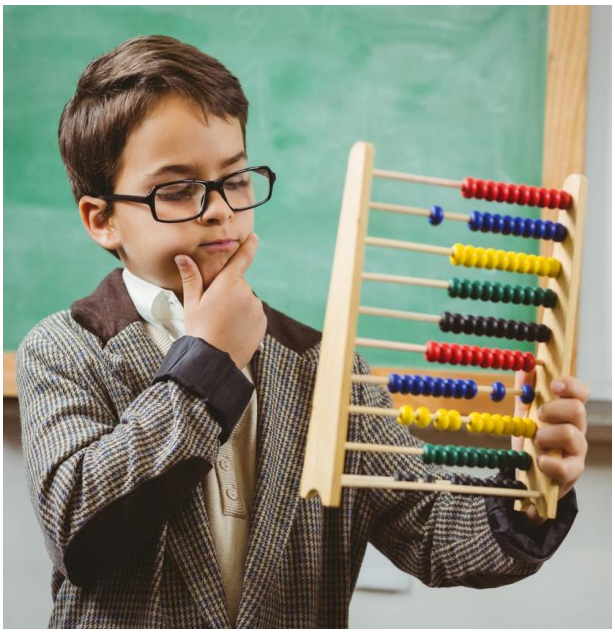
Output the word "**`Valid`**" if the input text is a valid room name; otherwise output the word "**`Invalid`**".

Sample input/output:

Input	Output
`W5-008 MacLab`	`Invalid`
`w8-105`	`Invalid`
`M95-889`	`Valid`

Problem C: Abacus

Balloon color: Green



Abaci (that is the plural of abacus) are counting frames widely used by kids to learn basic arithmetic. One of those kids, the boy in the picture to the right, has a request from you. He wants to represent a given number using an abacus by showing the values of different digits in the number. He wants the abacus to be drawn with vertical columns of beads. The used beads should be down the column; the unused should be up. Luckily, he counts to 999 only.

You will only draw three vertical lines of the abacus each line contains ten beads. The height of the columns (including the base and top) is 22 characters. There will be a single space separator between columns. No space before first column or after last column. You will use the following special characters in your drawing:

Character	ASCII Code	Usage
┴	193	base of a column
┬	194	top of a column
│	179	an empty unit of the column
╪	216	a bead in the column
─	196	a separator between columns in the top or base

(Those characters were used, before modern graphical computers, to print tables. USE THEM WITH RESPECT PLEASE).

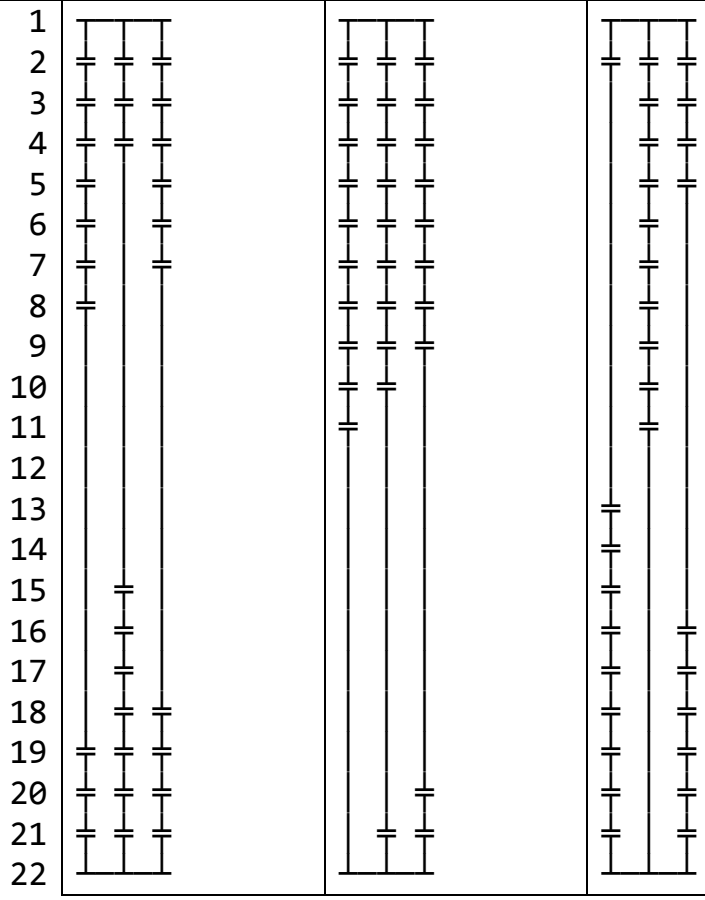
Input format:

A single integer N ($1 \leq N \leq 999$).

Output format:

The representation of N in a vertical 3-line abacus.

Sample input/output:

Input	374	12	906
Output (line numbers are displayed)			

Problem D: Unsocial

Balloon color: Red

Hurray!! Unsocial People Cinema (UPC) is now open. Unsocial people may now enjoy watching their favorite movies while maintaining their aloneness.

The cinema contains one row of chairs. Every unsocial person chooses a chair that is as far as possible from all previously seated viewers. Chairs are numbered (1 to N, where N is the number of chairs) a newcomer will choose his chair as follows:

1- The chair that is farthest away from a used chair is chosen,
2- If two (or more) chairs are equally far from the nearest used seat, the chair that is farther from the other side is chosen,
3- If two (or more) chairs are still equal, the chair with lower index is chosen.

Suppose UPC contains 7 chairs and the first person sits at chair 3. The six following persons will sit as follows:

	1	2	3	4	5	6	7
The 2nd person will choose chair 7 (the farthest)			🧍				↓
The 3rd person will choose chair 1 (Both chairs 1 and 5 are equal distance from nearest used seat but chair 1 has no used seat from other side (condition 2)	↓		🧍				🧍
The 4th person will choose chair 5 (the farthest)	🧍		🧍		↓		🧍
The 5th ,6th and 7th persons will choose chair 2, 4 and 6 respectively (all chairs are equal; lowest index is used)	🧍	↓	🧍	↓	🧍	↓	🧍

Given the number of chairs in the row and the chair used by first person, find where a certain late comer will be seated.

Input format:

The input consists of 3 integers separated by single spaces: N ($1 \le N \le 100{,}000$), F ($1 \le F \le N$) and T ($1 < T \le N$) where N is the number of chairs in the row, F is the chair used by the first person and T is the order of arrival.

Output format:

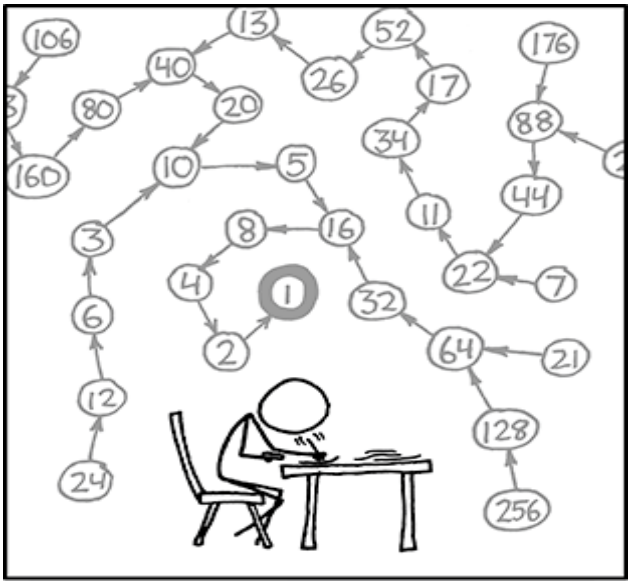
Output a single integer specifying the chair used by the T^{th} person.

Sample input/output:

Input	Output
7 3 4	5
15 5 8	8
20 1 20	19
1000 500 100	898

Problem E: Collatz Conjecture

Balloon color: Purple



Collatz Conjecture (named after the German mathematician Lothar Collatz, who introduced it in 1937) is a simple, yet not-proved, statement. Lothar stated that: Pick any positive integer. If even divide by 2; if odd multiply by 3 and add 1. Repeat and eventually you will get 1 (*before your friends think you are boring and stop calling you*).

Formally, for any positive integer N, define the series as:

$$C_1 = N;\ For\ n = 2,3,4,\ldots: C_n = \begin{cases} C_{n-1}/2, & if\ C_{n-1}\ is\ even \\ 3C_{n-1} + 1, & if\ C_{n-1}\ is\ odd \end{cases}$$

For a finite integer m: $C_m = 1$.

Examples of Collatz series for some values of N are given below:

N	$C_1, C_2, C_3, \ldots, 1$
10	10, **5**, 16, 8, 4, 2, 1
17	17, 52, 26, **13**, 40, 20, 10, 5, 16, 8, 4, 2, 1
39	39, 118, 59, 178, 89, 268, 134, 67, 202, 101, 304, 152, 76, **38**, 19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

The first element in the Collatz series of N that is less than N is called the gilde of N. The glide of each N is underlined in the above examples. Your task is to calculate the glide for any given integer N.

Input format:

Each input case is a single integer value N.

Output format:

For each input case, output the glide of N.

Sample input/output:

Input	Output
10	5
17	13
39	38

Problem F: IComparer Interface

Balloon color: Yellow

Generic programming using interfaces, such as `IComparer`, is a style of programming where code is written in terms of types *to-be-specified-later*. In line with that, `IComparer` interface allows programs to test ordering between two variables regardless of their types or how they are actually ranked. The standard behavior of method `Compare` in this interface is:

```
int IComparer::Compare(obj1, obj2) {
     if obj1 > obj2
          return 1
     else if obj1 < obj2
          return -1
     else
          return 0
}
```

You need to implement `IComparer::Compare` for numbers in scientific notation: numbers in the form $d \times 10^m$ where $1 \leq abs(d) < 10$.

Input format:

Each input contains two numbers in scientific notation expressed as $d_1\ m_1\ d_2\ m_2$ where d_1 and d_2 are decimal values such that $1 \leq abs(d_1), abs(d_2) < 10$ and m_1 and m_2 are integers such that $-10^8 \leq m_1, m_2 \leq 10^8$.

Output format:

Output a single value of 1, 0, or -1 by calculating `IComparer::Compare(`$d_1 \times 10^{m1}$`,` $d_2 \times 10^{m2}$`)` as described above.

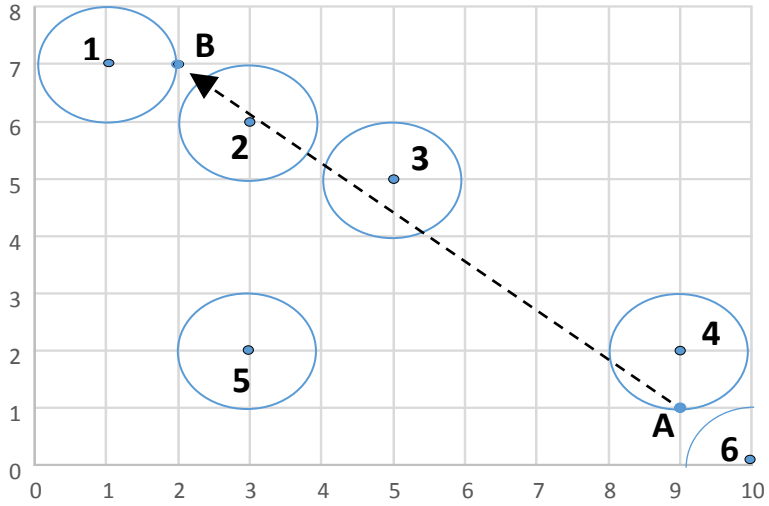
Sample input/output:

Input	Output
3.2 3 -3.2 3	1
-3.7 -40000000 -3.6 -40000000	-1
9.99 310 9.99 310	0
1.7 900 -2.3 1000	1
-1.7 900 -2.3 1000	1

Problem G: Football

Balloon color: White

Effective ball passing is essential for every football team. A team member needs to pass the ball to other teammates without being intercepted by opponent team. In order to ensure the ball is not intercepted, no opponent player should be close enough to catch the ball while it is passed. The ball is passed through the straight line between the two players. An opponent player can intercept the ball if he is at distance of one unit or less from any point in the ball path. The opponent player can also intercept the ball if he is at distance of one unit or less from the player who should receive the ball. However, if the opponent player is very close to the player who currently holds the ball (again one unit or less) he will not have the chance to intercept the ball.



In the left figure, if the ball is passed from player A to player B, the opponent players 1, 2 and 3 will be able to intercept the ball. Players 4, 5 and 6 are not in a position to intercept the ball.

Given the locations of players in both teams, you need to find how many teammates can receive the ball from a certain player (who currently holds the ball). Remember that two or more players may be at the same location.

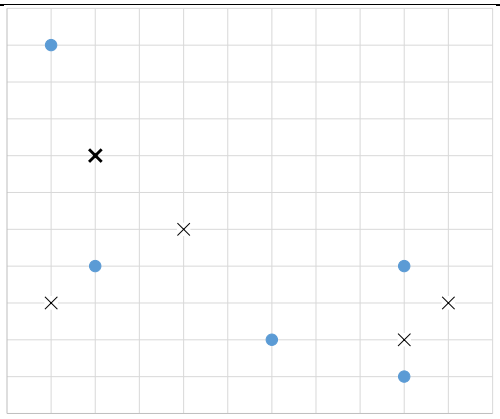
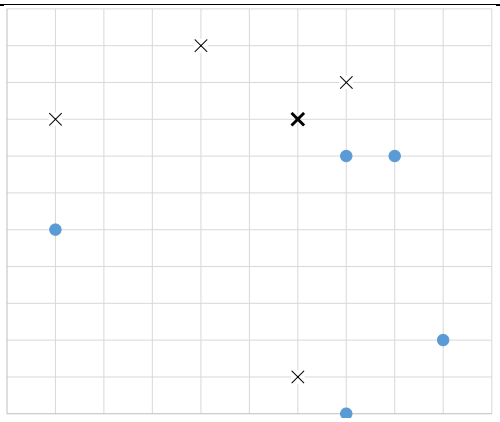
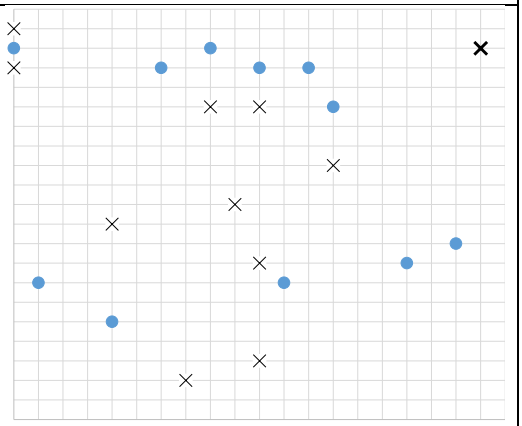
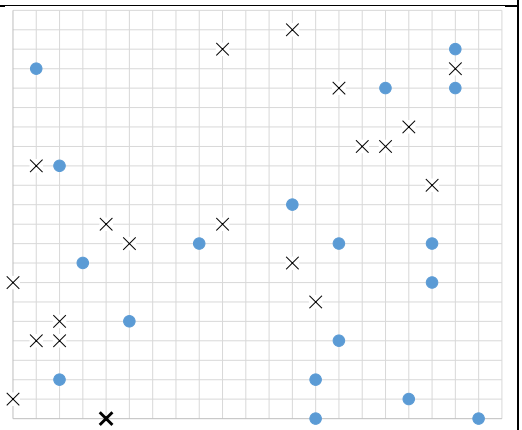
Input format:

The input starts with an integer N, where $2 \le N \le 20$, that specifies the number of players in each team. The x,y coordinates of the N players of the 1st team then the x,y coordinates of the N players of the 2nd team follow (i.e. a total of $4N$ integers) where $0 \le x,y \le 300$. The 1st player in the 1st team currently holds the ball.

Output format:

Output the number of players in the 1st team who can receive the ball from the 1st player in the team without being intercepted by a player from the 2nd team. This should be a number P, where $0 \le P \le N-1$.

Sample input/output:

Input	Output	Field layout team1:× (player1 bold), team2:●
5 2 7 9 2 10 3 4 5 1 3 1 10 9 1 6 2 2 4 9 4	1	
5 6 8 1 8 6 1 4 10 7 9 7 7 8 7 1 5 7 0 9 2	3	
11 19 19 10 8 0 20 9 11 10 16 13 13 4 10 8 16 7 2 0 18 10 3 8 19 13 16 4 5 1 7 10 18 6 18 16 8 11 7 18 9 0 19 12 18	4	
20 4 0 0 7 17 15 15 14 12 8 1 4 1 13 14 17 19 18 16 14 12 20 0 1 18 12 2 5 9 19 2 4 9 10 13 6 4 10 5 9 8 9 17 1 16 17 13 0 12 11 18 7 12 8 20 0 18 9 2 13 9 19 2 2 14 4 1 18 19 17 3 8 19 19 13 2 14 9 5 54	4	

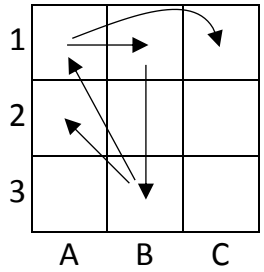
Problem H: Circular Formula

Balloon color: Pink

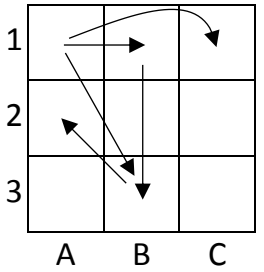
In the scoreboard in front of you, **"Solved"** is the count of green cells for each contestant, **"Time"** is the total submission time for all correct problems in addition to penalties while **"Rank"** is an ordering based on **"Solved"** and **"Time"**. Having such calculated cells is a powerful feature of spreadsheets. However, an important precaution has to be considered: AVOID CIRCULAR REFERENCE. A circular reference occurs when the formula of a cell references the cell itself directly or indirectly. We need your assistance to detect circular references. Samples of circular references are shown below.

Samples:

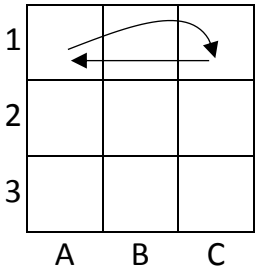
Each arrow shows a reference in formula from calculated cell to referenced cell. Cell names are column/row coded.



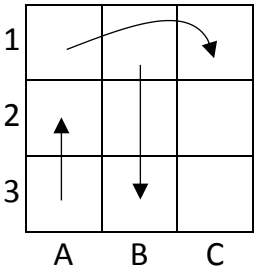
Formulae:
A1=B1+C1
B1=B3+1
B3=A1*A2
Circular reference:
A1, B1, B3



Formulae:
A1=B1+C1+B3
B1=B3+1
B3=2*A2
Circular reference:
None



Formulae:
A1= C1+1
C1=A1-1
Circular reference:
A1, C1



Formulae:
A1=3*C1
B1=B3+1
A3= A2/2
Circular reference:
None

Input format:

In the input, for simplicity, cells are numbered (i.e. not column/row named). The input starts with the total number of cells in the spreadsheet N ($2 \le N \le 100$). N lines follow. The i^{th} line (of these N lines) starts by an integer K ($0 \le K \le N-1$) specifying the number of cells referenced by the formula of cell number i followed by the numbers of these K cells. Cell numbers are 0-based (i.e. 0 to $N-1$).

Output format:

For each input case, output the lowest index of all cells involved in a circular reference. If no circular reference exists, output "**None**".

Sample input/output:

Input	Output
5 0 1 4 2 0 1 0 2 2 3	1
5 3 1 2 4 1 4 0 0 1 3	None
2 1 1 1 0	0
6 1 2 1 5 0 0 1 3 0	None

Problem I: Decryption Order

Balloon color: Black

BY READING THE BELOW MESSAGE, YOU WILLINGLY AND KNOWINGLY ACCEPT TO JOIN THE ULTIMATE PROGRAMMING CLIQUE. THIS MESSAGE WILL SELF-DESTRUCT IN FIVE HOURS.

An encrypted message is received by UPC commander. The encryption method is known: The characters of the original message are written diagonally in a square array. The characters are then re-assembled diagonally in the other direction. UPC commander orders you to decode the message and receive a generous grant (*the UPC prize*).

Example:

Original message "**decryption order**" is stored diagonally to the array as below.

1 ↙	2 ↙	3 ↙	4 ↙	
[1] d	[2] e	[4] r	[7] t	↙ 5
[3] c	[5] y	[8] i	[11]	↙ 6
[6] p	[9] o	[12] o	[14] d	↙ 7
[10] n	[13] r	[15] e	[16] r	

The letters from the array are read diagonally as below to get the encrypted message: "**tr eiddyorcoeprn**"

4 ↘	3 ↘	2 ↘	1 ↘	
5 ↘	[7] d	[4] e	[2] r	[1] t
6 ↘	[11] c	[8] y	[5] i	[3]
7 ↘	[14] p	[12] o	[9] o	[6] d
	[16] n	[15] r	[13] e	[10] r

Input format:

The input is the encrypted message in one line of text T ($4 \le length(T) \le 10{,}000$) and $length(T)$ is a perfect square.

Output format:

Output the decryption of text T (i.e. the original text that was scrambled to get T).

Sample input/output:

Input	Output
oced	code
[p1u2]c80	upc[2018]
tr eiddyorcoeprn	decryption order
virheaanoigeomgtlgnrfmorp	gatherinloveofprogramming

Problem J: Number Seven

Balloon color: Blue

The number seven certainly has a deep spiritual significance. There are 7 days, 7 Heavens, 7 Earths, 7 winners in UPC,… and the list goes on.

We want to count the numbers in which the fascinating digit 7 appears (let's call them fascinating numbers). *The answer is infinitely many. Are we done?* No wait. We will have a ceiling for our search. The problem now is to count fascinating numbers less than or equal to a certain given number. The strategy we developed to perform this spiritual task is:

Numbers of 1 digit (i.e. 0 to 9) contain 1 fascinating number which is 7 itself.
Numbers of 2 digits (i.e. 10 to 99) are formed by appending digits 1, 2, 3,…,9 to numbers of 1 digit. When we append any number other than 7 we get the same count of fascinating numbers for each appended number (i.e. 10 to 19 contain 1 fascinating numbers). When we append 7 (i.e. 70 to 79) then all the 10 numbers are fascinating. Thus:
Fascinating numbers of 2 digits = 8×(Fascinating numbers of 1 digit) + (All numbers of 1 digit) = 8×1 + 10 = 18.
Fascinating numbers from 0 to 99 = 18 + 1 = 19.
Fascinating numbers of 3 digits can be found from Fascinating numbers of 2 digits using the same logic and so on.
These partial counts can be used to find fascinating numbers in any range [1,N].
Suppose N =456 then divide the range as:
[1,456]=[1,99] ∪ [100,199] ∪ [200,299] ∪ [300,399] ∪ [400,409] ∪ [410,419] ∪ [420,429] ∪ [430,439] ∪ [440,449] ∪ [450,456].
Counts of fascinating numbers in the subranges are directly obtainable from the partial counts calculated above.

Input format:

The input is a single integer N ($1 \le N \le 2 \times 10^9$).

Output format:

Output the count of fascinating numbers in [1,N].

Sample input/output:

Input	Output
279	55
1743	448
1000800	468792
1986543765	1217755837