Gulliver and Telephone

From the Lilliput land, Gulliver the traveler is now in the land of Giants where everyone is atleast 20 times taller than him. He has now made one of the Giant's home as his home. And is now helping Giant with his chores.

Every day the Giant makes a telephone call to his friend and Gulliver is supposed to connect the call. Giant has a huge telephone box, with 0 to 9 digits. Gulliver is to hop on each digit and make the call that the giant wants him to make.

Find the sequence of moves that Gulliver has to take, to make the call, in the minimum number of moves/jumps. You can return any path that does so.

Dial-Pad of Telephone looks like

0	1	2
3	4	5
6	7	8
9		

On Dial-Pad, we start at position 0,0.
We may make the following moves (i.e., priority as follows):
'U' moves our position up one row, if the position exists on the dial-Pad;
'D' moves our position down one row, if the position exists on the dial-Pad;
'L' moves our position left one column, if the position exists on the dial-Pad;
'R' moves our position right one column, if the position exists on the dial-Pad;
'-' adds the character dialPad[r][c] at our current position (r, c) to the answer

Input/Output

Input	Output	Comments
0402221234	**-DR-UL-RR---L-R-DLL-R-**	**INPUT FORMAT:** - A number in string format. **Explanation:** **For '0':** start at '0' no moves needed, print '-' **For '4':** start at '0' move DOWN reaches '3', print 'D' move RIGHT reaches '4', print 'R' and we reached required '4', print '-' **For '0':** start at '4' move UP reaches '1', print 'U'

		move LEFT reaches ‘0’, print ’L‘ and we reached required ‘0’, print ‘-‘ and so on and so forth. We will get the required output:
987654321	**DDD-URR-L-L-URR-L-L-URR-L-**	As Above

Ken and his blocks

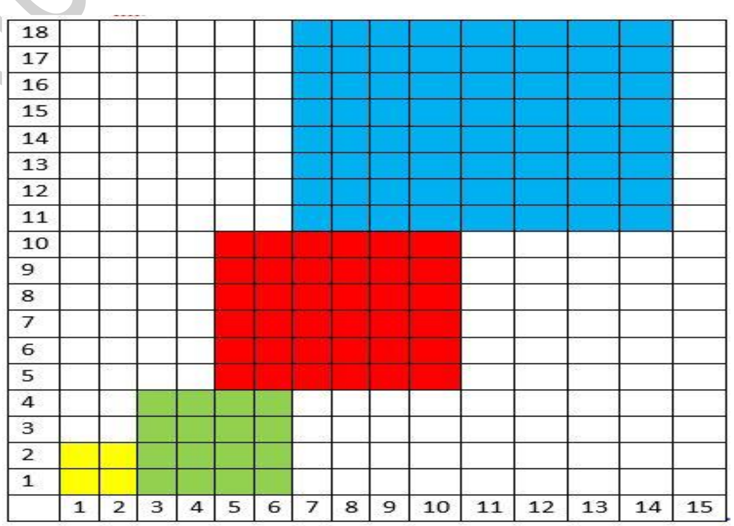
Little Ken is playing building blocks. As he is too young to play he ends up dropping the blocks than actually building. Assuming all blocks are cubes. Assuming the floor to be the number line (x-axis) , he drops the blocks in the order they are given(by us) .The j-th square dropped (positions[j] = (left,length_of_side)) is a cube with left-most point being positions[j][0] and sidelength positions[j][1].

The block is dropped with the bottom edge parallel to the number line, and from a higher height than all currently landed blocks. We wait for each block to stick before dropping the next.

The blocks are sticky on their bottom surface and will remain fixed to any positive length surface they touch (either the number line or another block). Blocks dropped adjacent to each other will not stick together prematurely.

Return a list hts of heights. Each height hts[k] represents the current highest height of any block we have dropped, after dropping blocks represented by hts[0],hts[1], so on ..hts[k].

Input/Output

Input	Output	Comments
4 1 2 3 4 5 6 7 8	2 4 10 18	INPUT FORMAT: 1st line – no of blocks (N) Following 'N' lines – position and side length of block OUTPUT FORMAT: Height of the building after each block dropped. EXPLANATION: 

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>3
1 2
3 4
7 2</td><td>2 4 4</td><td>
<table>
<tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
</table>
</td></tr>
</table>

NOTE: Colors used for better understanding.

Pair Square

In an array Arr with positive integers, when sum of two adjacent numbers form a perfect square you have found, a pair square.

Given an array, find the number of ways in which the numbers in the array can be arranged such that all pairs form perfect squares.

Two permutations of Arr1 and Arr2 differ if and only if there is one index j such that Arr1[j] != Arr2[j]

Input/Output

Input	Output	Comments
33 1 3	2	INPUT FORMAT: A line of integers. OUTPUT: Number of Permutations possible. EXPLANATION: PERMUTATION 1: 33 3 1-> 33+3, 3+1 -> Perfect Squares 1 3 33-> 1+3, 3+33 -> Perfect Squares 3 1 33-> 3+1 -> perfect square, 1+33 -> not a perfect square.
1 3 6 10	2	EXPLANATION: PERMUTATION 1: 1 3 6 10-> 1+3, 3+6, 6+10 -> Perfect Squares 10 6 3 1-> 10+6, 6+3, 3+1 -> Perfect Squares 3 1 33-> 3+1 -> perfect square, 1+33 -> not a perfect square.
2 2 2	1	EXPLANATION: PERMUTATION 1: 2 2 2-> 2+2, 2+2-> Perfect Squares We can't have any other combination