"Cube Square

Tom is very fond of cube and square numbers, i.e. numbers like 1, 4, 8, 9…

Given an integer N, count the number of integers from 1 to N that Tom likes.

In other words, find the number of such x that x is a cube of a positive integer or a square of a positive integer (or both a cube and a square simultaneously).

Note

While calculating cubes and squares, if any number is a cube as well as a square, then consider its existence at once only, i.e. square of 1 is 1 and cube of 1 is 1, so consider the count of 1 only once (not twice).

Function Description

In the provided code snippet, implement the provided cubeSquare(...) method using the variables to print the count of squares and cubes from 1 to N (inclusively), for each test case. You can write your code in the space below the phrase “WRITE YOUR LOGIC HERE”.

There will be multiple test cases running so the Input and Output should match exactly as provided.

Input Format

The first line of input contains a single integer T denoting the number of test cases.

Next T lines of input contain a single integer denoting N.

Sample Input

3 --denotes T

10 --denotes N

25 --denotes N

1 --denotes N

Constraints

1<= T <=10000.

1<= N <=1000.

Output Format

The T lines of output contain the count of squares and cubes from 1 to N (inclusively) for every test case.

Sample Output

4

6

1

Explanation

For the first test case,

1, 4, 9 are the squares of 1, 2, 3 respectively.

1, 8 are the cubes of 1, 2 respectively.

1 is the cube as well as the square so consider it only once.

Hence, the output is 4.

For the second test case,

1, 4, 9, 16, 25 are the squares of 1, 2, 3, 4, 5 respectively.

1, 8, are the cubes of 1, 2 respectively.

All the numbers are different, so the output is 6.

For the third test case,

1 is the square of 1

1 is the cube of 1.

Both are counted only once, so the output is 1."

internal class Program

{

// Function to check if the

// number is a perfect square

static bool isSquare(int num)

{

int root = (int)Math.Sqrt(num);

return (root \* root) == num;

}

// Function to check if the

// number is a perfect cube

static bool isCube(int num)

{

int root = (int)Math.Pow(num, (1.0 / 3.0));

return (root \* root \* root) == num;

}

// SubFunction to count the number of squares and cubes

static int cubeSquare(int N)

{

int CScount = 0;

for (int i = 1; i <= N; i++)

{

if (i < 2)

{

CScount++;

}

else

{

// If a number is square

//int j = i;

//int sqrt = (j+(i/j)) / 2;

int sqrt = (int)Math.Sqrt(i);

if ((sqrt \* sqrt) == i)

{

CScount++;

}

// Else if the number is cube or not

else

{

int croot = (int)Math.Pow(i, (1.0 / 3.0));

if ((croot \* croot \* croot) == i)

CScount++;

}

}

}

return CScount;

}

static void Main(string[] args)

{

Console.WriteLine("Sample Input");

int T = int.Parse(Console.ReadLine());

int N = int.Parse(Console.ReadLine());

Console.WriteLine("Sample Output");

Console.Write(cubeSquare(N));

Console.ReadKey();

}

}