# Exercise 1 – Run up the stairs

You are climbing a staircase. The staircase consists of some number of flights of stairs separated by landings. A flight is a continuous series of steps from one landing to another. You are a reasonably tall athletic person, so you can climb a certain number of steps in one stride. However, after each flight, there is a landing which you cannot skip because you need to turn around for the next flight (which continues in a different direction).

You will be given the number of steps in each flight in a Integer[] flights. Element 0 of flights represents the number of steps in the first flight, element 1 is the number of steps in the second flight, etc. You will also be given an Integer StairsPerStride, which is how many continuous steps you climb in each stride. If it takes two strides to turn around at a landing, return the number of strides to get to the top of the staircase. You do not need to turn at the top of the staircase.

The staircase has between 1 and 50 flights of stairs, inclusive. Each flight of stairs has between 5 and 30 steps, inclusive. StepsPerStride is between 2 and 5, inclusive.

Examples:

Input: {15}, StepsPerStride: 2, Returns: 8

A simple staircase with 15 steps. In 7 strides, you've climbed 14 steps. However, you still have one step left, so you must use an additional stride to get to the top.

Input: {15, 15}, StepsPerStride: 2, Returns: 18

This time, there are two flights with a landing in between. 8 strides to get to the first landing, 2 strides to turn around, and 8 more strides to get to the top makes 8+2+8=18 strides.

Input: {5,11,9,13,8,30,14}, StepsPerStride: 3, Returns: 44

# Exercise 2 – Safest place in the galaxy

While en route to the 295th annual Galactic Music Awards on Bandoo, you find yourself unceremoniously yanked out of hyperspace and, according to your sensors, surrounded by N space bombs. Apparently caught in a trap laid out by some dastardly and unknown enemy, and unable to return to hyperspace, you must find the safest place in the vicinity to weather the detonation of all the space bombs. Your unseen opponent has constructed a cube-shaped space anomaly that you are unable to leave, so your options are limited to points within that cube.

Before the bombs explode (all simultaneously), you have just enough time to travel to any integer point in the cube [0, 0, 0]-[1000, 1000, 1000], both inclusive. You must find the point with the maximum distance to the nearest bomb, which your captain's intuition tells you will be the safest point.

Input:

The first line of the input file consists of a single number T, the number of test cases. Each test consists of single number N, the number of bombs, followed by 3\*N integers describing the positions of the bombs.

Output:

Output T integers, one per test case each on its own line, representing the square of distance to the nearest bomb from the safest point in the cube.

Constraints:

* T = 50
* 1 <= N <= 200
* All bombs coordinates will be in [0, 1000], both inclusive

# Exercise 3 – Adding this, adding that

[Test]

[TestCaseSource("Add\_Source")]

public AddResult Add\_UsingARecursiveAlgorithm\_ValuesAreAdded(byte[] f, byte[] s)

{

// Arrange

// Act

var result = AddRecursive(f, s);

// Assert

return new AddResult(f, s, result);

}

E.G.

Input : { 1, 1, 1 }, { 1, 1, 1 }

Result: {2,2,2}

Input : { 1, 1, 255 }, {0, 0, 1 }

Result: {1,2,0}

Conditions:

* You can assume inputs f & s are never null, and are always of the same length.
* The algorithm should be non-destructive to the inputs.
* The algorithm should be able to handle large input lengths, of a couple of thousand values, but the input will never be large enough to cause a stack overflow.

Solve with a recursive method.