# 2019 Spring Final Contest

## Problem 1: Longevity

In general, run-length encoding is a trivial compression mechanism. It replaces a count of repeated data with a value-count pair. In the special case of encoding binary data, the value can be omitted since the values must alternate. For example, the run-length encoding for the binary number **1110000101100** is the sequence (3, 4, 1, 1, 2, 2).

If the binary value of each number in the sequence are concatenated together we obtain a new binary number: **11 100 1 1 10 10**, which could, in turn, be run-length encoded.

The *longevity* of a number is the number of times a binary number can be run-length encoded and concatenated before the length of the bit sequence is reduced to two. For example, the longevity of **1110000101100** is eleven, as shown below.

```
1110000101100
  11100111010
    111011111
       111101
        10011
        11010
        10111
         1111
          100
          110
          101
          111
           11
```

Write a program to compute the longevity of several numbers. The first line of input will contain the number of binary numbers to process. Each binary number will appear on a line by itself without spaces. Binary numbers will contain at most 63 bits.

Each data set should produce one line of output indicating the longevity of the input number.

## Sample Input

```
3
1110000101100
1010101010
000000011111111111111
```

## Sample Output

```
11
7
2
```

1110000101100 = 0+3+2+0+0+3+1 = 9 + 2 = 11

1010101010 = 2+2+2+2+2 = 10

# 2019 Spring Final Contest

## Problem 2: Partition

The Competition Council on Tournaments (CCT) is obsessed with fairness and mathematical rigger. If there exists more than one way to do something, the CCT insists that all possible solutions must be considered and a complete mathematical model must be developed. For example, choosing a subset of programming contest problems based on the expected difficulty is a typical problem. The CCT has been challenged to divide contest problems into two groups. The first step is to create a mathematical model.

> A Partition divides a problem set (S) into subsets (S1 and S2), such that S = S1 ∪ S2, and ∅ = S1 ∩ S2.
>
> A Difference Partition (DP) of a set of problems is the absolute difference between the total expected difficulty of S1 and the total expected difficulty of S2.

Write a program to compute the minimum DP of several sets of contest problems.

The first line of input will contain a positive integer, the number of sets to process. Each set is specified by two lines. The first line contains a single positive integer, the number of problems in the set (N ≤ 100). The second line contains N positive integers, the expected difficulty of each problem (e ≤ 100), each separated by a single space.

Each data set should produce one line of output indicating the minimum DP for the contest problem set.

## Sample Input

```
3 test cases
1 problems in set
7 difficulty
4 problems in set
3 9 3 3 difficulty
4
1 5 11 6
```

## Sample Output

```
7
0
1
```

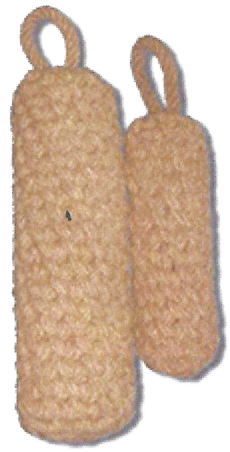
for (int i 0 → testcases)

problems in set

difficulty

# 2019 Spring Final Contest

## Problem 3: Fenders



Once upon a time there was a rope weaver who made realistic boat fenders used to keep boats from scraping against each other, docks or piles. As a professional member of the Conscientious Committee on Tying (CCT), the rope weaver's fenders always require exactly the same length of rope for a particular fender design. Ropes are never spliced as it subtly changes the shape of the fender and CCT members are facidious about authenticity. All CCT fenders are made from authentic hemp rope available only on 500' spools.

Given the length of rope required for a fender, and the number of fenders ordered, determine how many scrap rope pieces will be wasted in order to complete the order.

The first line will contain the number of data sets to process.

Each data set consists of the number of fenders ordered, followed by a single space, followed by the length in feet required to weave one fender.

Each data set should produce one line of output indicating the number of scrap pieces generated.

## Sample Input

```
4
15 100
3 200
3 166
5 167
```

1500
600
total = total - length
1400
rope = rope - length
&&!=0

## Sample Output

```
0
2
1
3
```

# Spring 2019 Final Contest

## Problem 4: Sum

A matrix is a 2-dimensional array of integers. A submatrix is any contiguous, rectalinear subset of a matrix; that is, the submatrix can be obtained by deleting any combination of rows and columns from the *edges* of the original matrix. The sum of a matrix is obtained by adding all elements of the matrix. A maximum submatrix is the submatrix with the largest sum.

Given a matrix, find its maximum submatrix.

The first line will contain the number of data sets to process.

Each data set consists of the number of rows and columns on a single line, followed by one line for each row of the matrix. Each line contains one value for each column of the matrix, each separated by one space, as shown below. Row and column numbers will be positive integers not more than 200.

Each data set should produce one line of output indicating the sum of the maximum submatrix.

## Sample Input

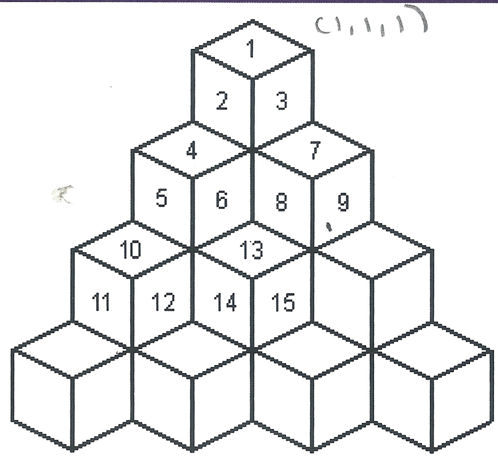
```
3    test cases
3 5    size
1 2 -2 3 -4
-2 1 3 0 1
3 -2 0 2 3
5 5    size
1 2 -2 3 -4
-2 1 3 0 1
3 -2 0 2 3
1 0 2 0 -1
0 -1 1 1 5
1 1    size
-1
```

## Sample Output

```
9
17
0
```

# 2019 Spring Final Contest

## Problem 5: Painter




Michael Angelo paints murals on steps and faces similar to the ones shown at the right. He uses the unpainted steps to reach the step or face he will paint next. After trying several alternative schemes, he decided to use the order as shown on the right.

For historical reasons, Michael identifies "cubes" using a 3D, 1-based cartesian coordinate system. The "origin cube" (1,1,1) is located on the bottom plane and is farthest away from the viewer. The topmost "cube" has coordinate (1,1,4). The rightmost cube has coordinate (4,1,1).

When each work is commissioned, Michael enters the commissioner's request for each step or face into a database using a code based on the "cube" coordinate and a letter indicating whether the mural should be painted on the top step (T), left face (L), or right face (R). Left and right are from the perspective of a viewer.

Given the number of the mural Michael should paint next, find its database code as described above.

The first line will contain the number of data sets to process.

Each data set consists of one line containing the size of the height of the steps and the mural number, with a single space between them.

Each data set should produce one line containing the database code as shown below.

## Sample Input

```
3  data sets
4 1  size of height, mural number
4 15
5 5
```

## Sample Output

```
1 1 4 T
2 2 2 R
1 2 4 L
```

# 2019 Spring Final Contest

## Problem 6: Farmer

Due to recent economic and atmospheric developments, farmer Schwarz decided to convert his failed corn crop into a public attraction to raise additional funds. By carefully chopping some of the corn he was able to create a labyrinth, allowing the public to wander at their leisure for an hourly rate. And thus he was able to keep his farm safe from the bank for another year.

Unbeknownst to him, alien worms were observing his fields and added a twist to secretly make watching the public more fun. The aliens added tunnels to the maze so that anyone entering one end of the tunnel is instantly, and harmlessly, sent to its other end, somewhat akin to teleportation.

Once the farmer's son discovered the tunnels and realized that the public was thrilled by the experience, he convinced his father to sell the farm and start a new business known as Amazing Schwarz's Child Worm Hole Maize Mazes.

Schwarz's son is worried that folks may get out of the maze too quickly so he needs a program to help determining the shortest path thru a maze. Given a description of an Amazing Schwarz's Child Worm Hole Maize Maze, find the minimum number of steps required to reach the exit.

The first line will contain the number of data sets to process.

Each data set begins with a line containing the width and height of the maze, with a single space between them, both at most 100.

Each row of the maze is given by a line of characters. Period (.) indicates the space is open. Maize is shown as an capital letter (X). Worm holes are indicated by lowercase letters. If a lowercase letter appears in a maze, it will appear exactly two times in the maze.

All edge locations will be maize-filled, except one entrance, on the left side, and one exit, on the right side.

Each data set should produce one line of output containing the minimum number of steps needed to reach the goal, if it is possible. If it is not possible to reach the exit, output the phrase "MONEY MAZE".

### Sample Input

```
3
10 5
XXXXXXXXXX
X..a.....X
........aX
X.........
XXXXXXXXXX
10 5
XXXXXXXXXX
X.....XbaX
...X.XXXXX
X..XaXb...
XXXXXXXXXX
6 3
XXXXXX
.aXaX.
XXXXXX
```

### Sample Output

```
7 STEPS
12 STEPS
MONEY MAZE
```