Melon Blaster

Introduction

In the game of archery there are 10 melons placed one after the other in a straight line. The objective of the archer is to shoot an arrow right through all 10 melons.

Scoring

A game of archery is scored as follows.

- Each archer takes 10 turns shooting at the melons. Every turn, he has 2 arrows to shoot.
- If the archer blasts through 3 melons with the first arrow, in the second attempt archer has a shot at the remaining 7 melons.
- If the archer manages to blast through 3 more the overall score for this turn would be scored at 6.
- If the archer manages to pierce through all remaining 7 melons the score would be 10 + whatever archer scores in the next try.
- If the archer manages to pierce through all 10 melons with a single arrow score is 10 + whatever archer scores in the next 2 tries.
- On the 10th turn is scored slightly different:
    - If the archer manages to score a perfect 10 with the first arrow , another shot will be available to the archer at 10 more melons.
    - If a perfect 10 is scored again, another turn will be presented to the archer to shoot through 10 melons again.
    - If the archer manages to score 10 with the first 2 arrows , one more shot at 10 melons is presented to the archer and whatever is scored there will be added to the overall score.

Challenge

Given a scoresheet of an archer, write a program to calculate the overall score. Below are some example scoresheets with expected results

Objectives

- Extensible, easily adaptable to future enhancements.
- Highly cohesive and loosely coupled.
- Good use of OOP concepts.
- Code should be clean. Avoid code smells.
- Can easily be unit tested. Write unit tests.

Examples

Given an archer with the following results:

- Turn 1
  - 1st arrow -> 4
  - 2nd arrow -> 2
- Turn 2
  - 1st arrow -> 3
  - 2nd arrow -> 4
- Turn 3
  - 1st arrow -> 7
  - 2nd arrow -> 3
- Turn 4
  - 1st arrow -> 10
  - 2nd arrow -> No need to shoot(arrow will be returned)
- Turn 5
  - 1st arrow -> 6
  - 2nd arrow -> 0
- Turn 6
  - 1st arrow -> 0
  - 2nd arrow -> 5
- Turn 7
  - 1st arrow -> 9
  - 2nd arrow -> 0
- Turn 8
  - 1st arrow -> 2
  - 2nd arrow -> 1
- Turn 9
  - 1st arrow -> 5
  - 2nd arrow -> 2
- Turn 10 (can have a maximum of 3 arrows)
  - 1st arrow -> 10
  - 2nd arrow -> 10
  - 3rd arrow -> 5

The archer scores **104** points.

A second archer has the following results:

- Turn 1
  - 1st arrow -> 4
  - 2nd arrow -> 6
- Turn 2
  - 1st arrow -> 10
  - 2nd arrow -> No need to shoot(arrow will be returned)
- Turn 3
  - 1st arrow -> 0
  - 2nd arrow -> 10

- Turn 4
    - 1st arrow -> 3
    - 2nd arrow -> 4
- Turn 5
    - 1st arrow -> 3
    - 2nd arrow -> 1
- Turn 6
    - 1st arrow -> 0
    - 2nd arrow -> 5
- Turn 7
    - 1st arrow -> 9
    - 2nd arrow -> 0
- Turn 8
    - 1st arrow -> 2
    - 2nd arrow -> 1
- Turn 9
    - 1st arrow -> 5
    - 2nd arrow -> 2
- Turn 10 (can have a maximum of 3 arrows)
    - 1st arrow -> 3
    - 2nd arrow -> 2

This archer scores **97** points.

Input

The archer scoresheet is given as a text file containing a single row for each turn. Each turn is described by the number of melons the archer hits in that turn, separated by comma's.

For example, this is the scoresheet corresponding to the first example above:

```
4,2
3,4
7,3
10,
6,0
0,5
9,0
2,1
5,2
10,10,5
```

Delivery

Your solution should be a command-line application that takes as input one parameter, the name of the file containing the scores for a single game of archery, and writes to standard output the total score for the game.

Your solution must be started using a shell script called `run.sh` that accepts this parameter. After the calculation completes, your program should write it's output in the format shown below.

Example:

```
> ./run.sh sample-1.txt
Score: 104
```