**Name:**

**Advanced Programming in Java**

**Lab Exercise 1.13.2025**

1. **Find the Other Two Side Lengths**

Given the shortest side of a 30° by 60° by 90° triangle, find out the other two sides. Return the longest side and medium-length side in that order.

**Examples**

otherSides(1) ➞ [2.0, 1.73]

otherSides(12) ➞ [24.0, 20.78]

otherSides(2) ➞ [4.0, 3.46]

otherSides(3) ➞ [6.0, 5.2]

**Notes**

* 30° by 60° by 90° triangles always follow this rule: let's say the shortest side length is x units, the hypotenuse would be 2x units and the other side would be x \* square root of 3.
* The **results** in the **Tests** are rounded up to 2 decimal places.
* Return the **result** as an **array**.

1. **Crowded Carriage Capacity**

A train has a maximum capacity of n passengers overall, which means each carriage's capacity will share an equal proportion of the maximum capacity.

Create a function which returns the **index** of the first carriage which holds **50% or less** of its maximum capacity. If no such carriage exists, return -1.

**Worked Examples**

findSeat(200, [35, 23, 18, 10, 40]) ➞ 2

// There are 5 carriages which each have a maximum capacity of 40 (200 / 5 = 40).

// Index 0's carriage is too full (35 is 87.5% of the maximum).

// Index 1's carriage is too full (23 is 57.5% of the maximum).

// Index 2's carriage is good enough (18 is 45% of the maximum).

// Return 2.

**Examples**

findSeat(20, [3, 5, 4, 2]) ➞ 3

findSeat(1000, [50, 20, 80, 90, 100, 60, 30, 50, 80, 60]) ➞ 0

findSeat(200, [35, 23, 40, 21, 38]) ➞ -1

**Notes**

* If a train can hold **200** passengers, and has **5** carriages, then that means that each carriage can hold a maximum of **40** passengers each.
* All carriage numbers will be positive integers, which will be able to divide evenly.
* Remember to return -1 if no carriage is empty enough.

1. **Musical Instrument Note Ranges**

Musical instruments have a range of notes to play, some instruments having a much larger range than others.

Given the following ranges for the instrument, create a function that returns true if a given note is within a given instrument's range. Otherwise, return false.

| **Instrument** | **Range** |
| --- | --- |
| Piccolo | D4-C7 |
| Tuba | D1-F4 |
| Guitar | E3-E6 |
| Piano | A0-C8 |
| Violin | G3-A7 |

**Examples**

instrumentRange("Piccolo", "A3") ➞ false

instrumentRange("Violin", "G6") ➞ true

instrumentRange("Piano", "C8") ➞ true

**Notes**

* Tests will only include natural notes (i.e. you will only deal with letters and numbers, no special characters).
* The musical scale follows this pattern: ... A1, B1, C1, D1, E1, F1, G1, A2, B2 ...
* You don't need to worry about invalid inputs.

**4. Video Length in Seconds**

You are given the length of a video in minutes. The format is **mm:ss** (e.g.: "02:54"). Create a function that takes the *video length* and return it in **seconds**.

**Examples**

minutesToSeconds("01:00") ➞ 60

minutesToSeconds("13:56") ➞ 836

minutesToSeconds("10:60") ➞ -1

**Notes**

* The video length is given as a string.
* If the number of seconds is **60 or over**, return -1 (see example #3).
* You may get a number of minutes over 99 (e.g. "121:49" is perfectly **valid**).