Project 1: Shortest Path

General Instructions for all projects

- *You must submit a document PDF via BB to explain your approach. You can submit code in the document, or via Github link in the document. You can create your document in Word, Google Docs, Slide Deck, etc, but only submit the output PDF.*
- *You need to Submit **AND** demo your program during office hours to receive a grade and/or feedback.*
- *For team projects a SINGLE submission (from any student) and a SINGLE demo is needed.*

Project 1 Introduction

In this project, we will be using the toy water pitcher problem to expand on our understanding of the "search" problem. The learning objectives are:

(i) Search problem
(ii) Dynamically creating a graph, when there are no nodes or edges in sight.
(iii) A* algorithm, heuristic, and lower bound

Input and Output

You are given a text file with 2 lines. First line has a variable number of integers, comma separated. Second line has an integer. There are no comments etc. in the file. *(Don't bother trying to handle negative numbers, fractional numbers, alphanumeric test cases etc. I trust you know how to do that, and this class is not focused on that.)*

Input File Structure

input.txt

```
1,4,10,15,22    // Represents the capacities of the different water pitchers
                // You also have access to a virtual "infinite" capacity pitcher
181             // Represents the Target quantity
```

Sample Files

```
>> cat input1.txt
2,5,6,72
143

>> cat input2.txt
3,6
2

>> cat input3.txt
2
143

>> cat input4.txt
2,3,5,19,121,852
11443
```

Output

Output is a single number which represents the number of steps.

To Do

- Implement a program that reads this input file and calculates the shortest path (number of steps) from initial state (all pitchers are empty) to the final state (where the "infinite" capacity pitcher has the target quantity).
- Any kind of water movement (from any pitcher to any other pitcher) represents one step.
- Implement an informed search (A*) for this problem.
- Write test cases (JUnit or equivalent) to ensure to make sure your program works. Unit test code is also included in the grading.

Things to Consider

1. In your write up you need to explain your lower bound. If you don't have a lower bound, that means you don't have A* algorithm.
2. If there is NO path, then simply return -1.