Problem-Solving Session Rules

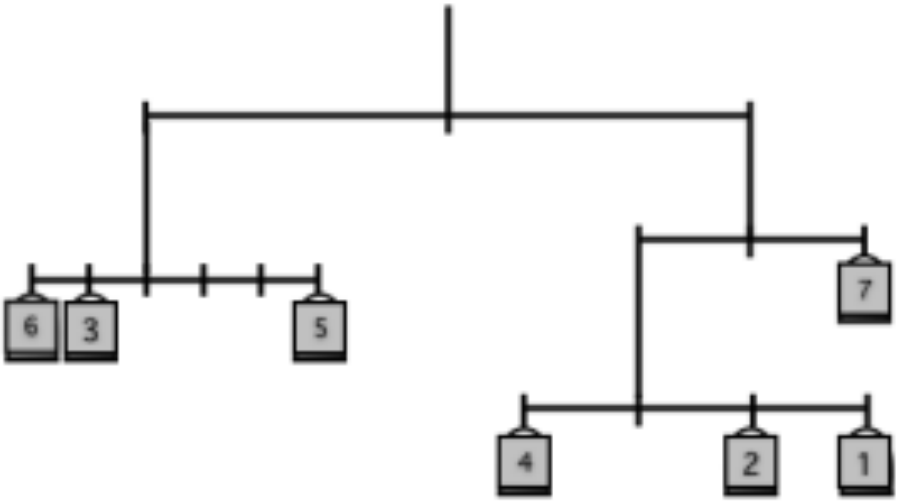
- Each team member must contribute to answering all the questions from the problem-session. You may lose up to 20% of your lab grade if you don't contribute.
- If a question requires you to write code, work with your teammates to write the code in this document (do not use PyCharm nor any IDE).
- Before leaving the meeting, make sure you download this document with your answers.
-
- You will probably need it later for the lab implementation.
- Check with your SLI or instructor your answers before leaving the meeting.

Do not forget to enter your name in the team members section.

Team Members

Steve Gao, sg2369@rit.edu
Qiwen Quan, qq5575@rit.edu

1.

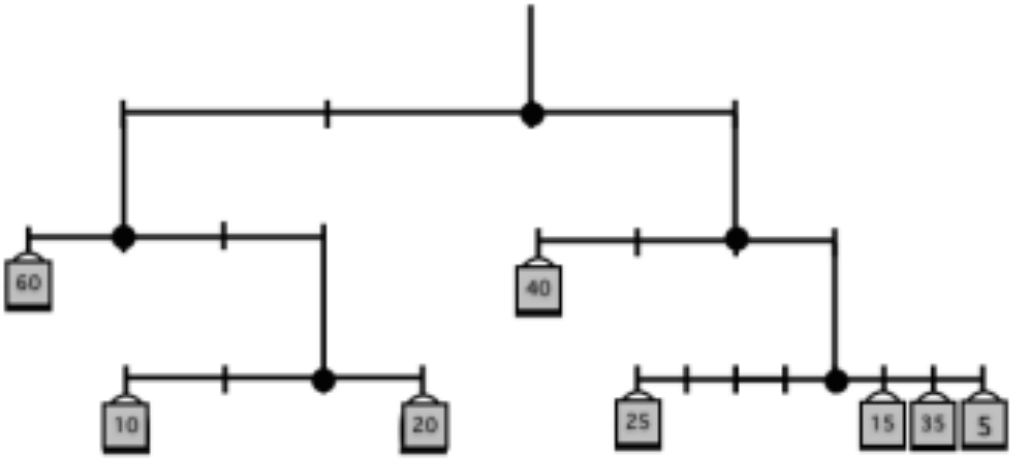


6 + 3 + 5 = 14 7 + 7 = 14

4 + 2 + 1 = 7*1 = 7 7*1 = 7

6 * 2 + 3 * 1 = 15 3 * 5 = 15 4 * 1 = 4 2*1 + 1*2 = 4

It is balanced



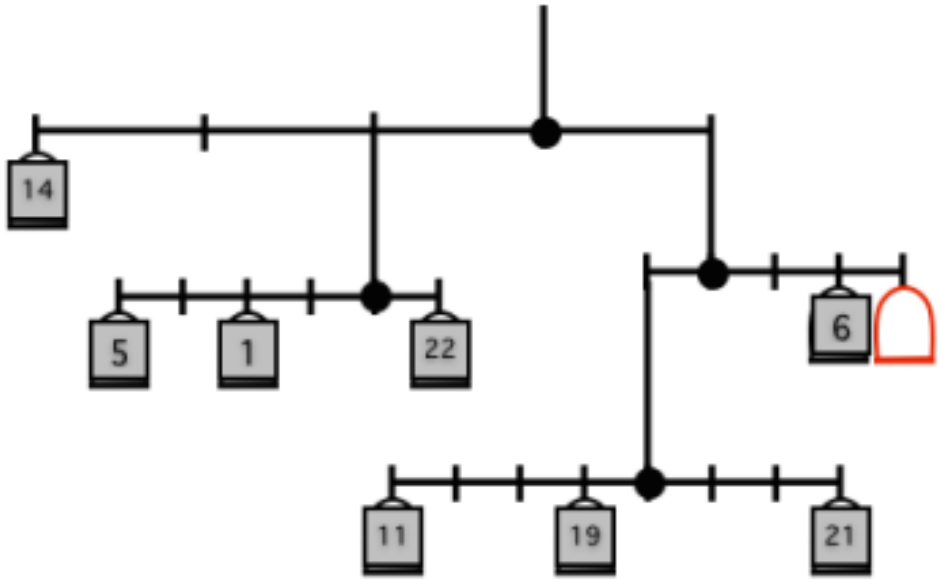
Not balanced

(60 + 30) * 2 = 180 (40 + 80) * 1 = 120

60*1 = 60 (10+20)*2 = 60 40 * 2 = 80 80 * 1 = 80

10*2 = 20 20*1 = 20 25*4 = 100 15*1+35*2+5*3 = 100

Q2



11 + 19 + 21 = 51 -> 51 * 1 = 51 6*2 + x*3 = 51 x = **13**

28 + 14*3 = 70 11+19+21+6+13 = 70

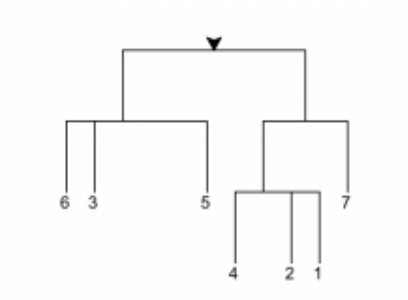
Q3
__slots__ = ‘weights’, ‘distance’
Weights: A collection of weight objects. Assume position of this weight is recorded in the Weight object.
Distance: The distance from the hanging point of this beam to the center point of the beam that hangs this beam. 0 if this beam is at the top.

Q4

```
def weight(self):
        weight = 0
        for w in self.weights:
                weight += w.weight()
        return weight
```

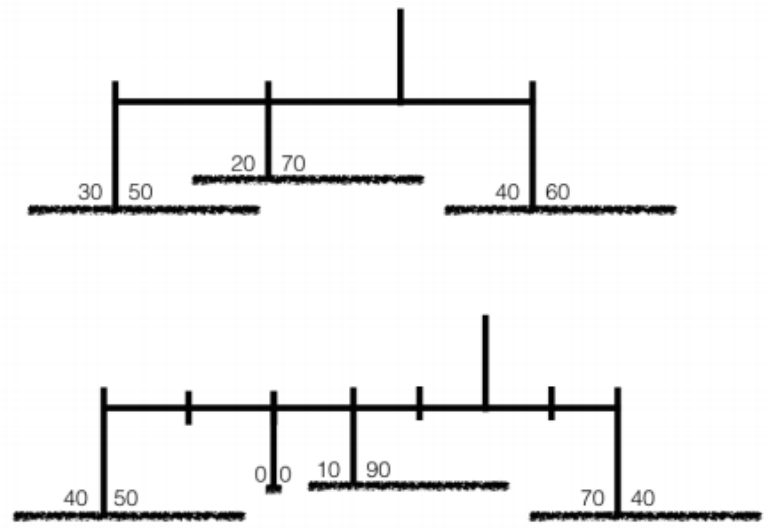
Q5

5. We would also like to draw balance puzzles from a text description. When drawn, it might look something like this:



Importantly, we do not want different beams to overlap. To make sure this happens, we will have to scale each beam left-to-right to separate the things that hang from it so that they do not overlap. In fact, we would like to leave a gap of at least 20 pixels between any two adjacent things that hang below. To simplify things, we will **not** allow any vertical overlaps such as appear in the puzzle in question 2.

In each of the following partial balance puzzles, the numbers on top of each lower beam represent the number of pixels from the center of that beam to its farthest extent (left and right). They have been drawn at different heights just for clarity here, in practice they would all be drawn at the same height. The zeros in the second balance refer to a weight, that we assume has 0 width.



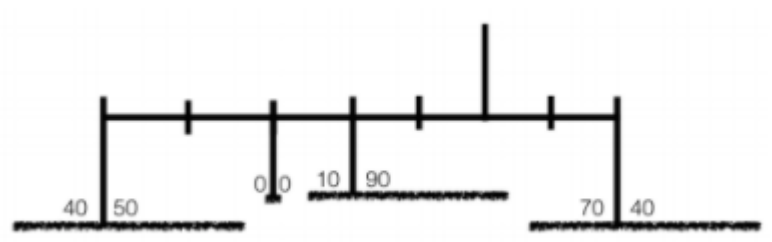
(a) For each balance, what should be the scale factor (pixels per tick along the beam) so that there is at least 20 pixels of space between adjacent lower beams/weights?

(b) For each balance, how far left and right will it extend relative to its hanging point? You should include the width of those things hanging from it as well.

Puzzle 1. (a) Scale factor: max((50+20+20),(70+40+20)/2) = 90

b)
Total width (extension) 30 + (90*3) +60 = 360
Far left = 30 + (90*2) = 210
Far right = (90 * 1) + 60 = 150

Puzzle 2. (a) Scale factor: max((50+0+20)/2, (0+10+20), (90+70+20)/4) = 45



(b)
Width (Extension) : 40 + (45 * 7) + 40 = 395
Far left: 40 + 45*5 = 265
Far right: 45*2 + 40 = 130