

Frankfurt University of Applied Sciences
Programming Day Contest 2022

Benedikt Zundel

Decembter 8th 2022

Contents

1	Introduction	3
2	Problem Overview	4
3	Solutions	5
3.1	A: Coffee Cup Combo	5
3.2	B: Estimating the Area of a Circle	6
3.3	C: Fun House	6
3.4	D: Espresso!	7
3.5	E: Pea Soup and Pancakes!	8
3.6	F: Math Trade	8
3.7	G: Love Polygon	9
3.8	H: Santa Klas	9
3.9	I: Triangle Ornaments	10
3.10	J: Sacred Texts	10
3.11	K: Spelling Bee	11
3.12	L: Fifty Shades of Pink	11

1 Introduction

This document serves as a writeup for the Frankfurt University of Applied Sciences Programming Day Contest hosted by Prof. Dr. Logofatu on December 8th 2022. It contains an outline of the problems presented during the contest with information about difficulty¹ for revision and demonstrates a possible way of solving them (in Python).

A repository containing all the solutions contained in this document can be found at <https://github.com/bezunyl/frauas-programming-day-2022>.

¹The difficulty of the problems may change [1]. Given level corresponds to the difficulty viewed on 15th of December 2022.

2 Problem Overview

According to Kattis' Terms of Service [2] it is prohibited to copy any content displayed on their site, due to which I cannot display the problems and their descriptions in full within this document. However, the below table contains the name and difficulty to each questions of the contest and a link under which it can be found on the official Kattis webpage.

Tag	Name	Difficulty	Link
A	Coffee Cup Combo	1.5 Easy	https://open.kattis.com/problems/coffeecupcombo
B	Estimating the Area of a Circle	1.5 Easy	https://open.kattis.com/problems/estimatingtheareaofacircle
C	Fun House	1.9 Easy	https://open.kattis.com/problems/funhouse
D	Espresso!	2.1 Easy	https://open.kattis.com/problems/espresso
E	Pea Soup and Pancakes	2.2 Easy	https://open.kattis.com/problems/peasoup
F	Math Trade	3.9 Medium	https://open.kattis.com/problems/mathtrade
G	Love Polygon	n.a.	n.a.
H	Santa Klas	2.8 Medium	https://open.kattis.com/problems/santaklas
I	Triangle Ornaments	n.a.	n.a.
J	Sacred Texts	3.9 Medium	https://open.kattis.com/problems/sacredtexts
K	Spelling Bee	2.1 Easy	https://open.kattis.com/problems/spellingbee
L	Fifty Shades of Pink	2.1 Easy	https://open.kattis.com/problems/fiftyshades

3 Solutions

3.1 A: Coffee Cup Combo

In this problem we help a student figure out how many lectures she can stay awake for, based on the premise that she can only stay awake if she drinks one coffee per lecture. We are given the amount of lectures that the student attends on a single day and information for each lecture about whether or not the lecture hall contains a coffee machine to refill her two cups.

This leaves us with a fairly straightforward problem: We have to count how many lectures she can stay awake for by tracking the amount of full coffee cups she has. To do so, we view all possible situations. Either there is a coffee machine in the lecture hall, or there isn't (case 1 and 0, respectively). In case 1, we do not worry if she currently has any cups of coffee left, as she can fill both cups before the lecture, drink on during the lecture and finally fill both cups up again after the lecture. Therefore, we just increment the count of lectures she can stay awake for and reset her remaining full cups to two. In case 0 there are two branches we have to consider: Either she has at least one cup of coffee left (case 0_y) or both her cups are empty (case 0_n). In case 0_y we simply increment the count of lectures she does not sleep in and decrement the variable keeping track of how many cups of coffee she has left. Case 0_n is simple, as we do nothing.

```
1 n = input()
2 ls = input()
3
4 cups = 0
5 count = 0
6
7 for i in ls:
8     if i == "1":
9         cups = 2
10        count += 1
11
12    if i == "0":
13        if cups > 0:
14            count += 1
15            cups -= 1
16
17 print(count)
```

Listing 1: "Coffee Cup Combo Solution"

The above is an implementation of what was explained prior. First we receive the two lines of input as store them in variables, n being the amount of total lectures and ls being the presence of a coffee machine in the lecture hall, respective to its index. Two more variables are initialized to keep track of the amount of full coffee cups the student has (initialized to 0) and the amount of lectures she's awake in (the output), respectively. Following those two initializations is the beginning of a for loop which iterates over ls to read the presence of a coffee machine in each lectures hall. The decision tree explained

above is then implemented with if-statements. Finally, the output is printed to the terminal and the program terminates.

3.2 B: Estimating the Area of a Circle

```
1 import math
2
3 def realArea(r):
4     return math.pi * r ** 2
5
6 def approxArea(r, pm, pin):
7     return (pin / pm) * ((r * 2) ** 2)
8
9 def compute(r, m, c):
10    print("%f %f"%(realArea(float(r)), approxArea(float(r), int(m),
11    int(c))))
12
13 while True:
14     l = input()
15
16     if l == "0 0 0":
17         break
18
19     s = l.split()
20     compute(s[0], s[1], s[2])
```

Listing 2: "Estimating the Area of a Circle" Solution

3.3 C: Fun House

```
1 houseid = 1
2
3 def getCoordsOfChar(tiles, char):
4     for i, ts in enumerate(tiles):
5         if char in ts:
6             return (i, ts.index(char))
7
8 def startVelocity(tiles, ci):
9     if ci[0] == 0:
10        return (1, 0)
11    elif ci[0] == len(tiles) - 1:
12        return (-1, 0)
13    elif ci[1] == 0:
14        return (0, 1)
15    elif ci[1] == len(tiles[0]) - 1:
16        return (0, -1)
17
18 def addTup(a, b):
19    return(a[0] + b[0], a[1] + b[1])
20
21 def reverseTup(a):
22    return (a[1], a[0])
23
24 def inverseTup(a):
25    return (a[0] * -1, a[1] * -1)
```

```

26
27 def directionChange(tiles, pos, v):
28     if tiles[pos[0]][pos[1]] == "\\":
29         return reverseTup(v)
30
31     if tiles[pos[0]][pos[1]] == "/":
32         return inverseTup(reverseTup(v))
33
34     else:
35         return v
36
37 def printMap(tiles):
38     for ts in tiles:
39         print(" ".join(ts))
40
41 while True:
42     l = input()
43
44     if (l == "0 0"):
45         break
46
47     s = l.split()
48
49     w = int(s[0])
50     h = int(s[1])
51
52     tiles = []
53
54     for i in range(h):
55         ts = input()
56         tiles.append([t for t in ts])
57
58     entry = getCoordsOfChar(tiles, "*")
59     v = startVelocity(tiles, entry)
60     pos = addTup(entry, v)
61
62     while True:
63         if tiles[pos[0]][pos[1]] == "x":
64             tiles[pos[0]][pos[1]] = "&"
65             break
66
67         v = directionChange(tiles, pos, v)
68         pos = addTup(pos, v)
69
70     print("HOUSE %d"%(houseid))
71     printMap(tiles)
72
73     houseid += 1

```

Listing 3: "Fun House" Solution

3.4 D: Espresso!

```

1 def waterreq(order):
2     if len(order) == 1:
3         return int(order)
4     else:

```

```

5         return int(order[0]) + 1
6
7 l1 = input().split()
8 n = int(l1[0])
9 s = int(l1[1])
10
11 tank = s
12 count = 0
13
14 for i in range(n):
15     l = input()
16
17     if (tank - waterreq(l) < 0):
18         tank = s - waterreq(l)
19         count += 1
20     else:
21         tank -= waterreq(l)
22
23 print(count)

```

Listing 4: "Espresso!" Solution

3.5 E: Pea Soup and Pancakes!

```

1 n = int(input())
2 cond = False
3
4 for i in range(n):
5     c = int(input())
6     name = input()
7     food = []
8
9     for j in range(c):
10        f = input()
11        food.append(f)
12
13    if ("pea soup" in food) and ("pancakes" in food):
14        print(name)
15        cond = True
16        break
17
18 if cond == False:
19    print("Anywhere is fine I guess")

```

Listing 5: "Pea Soup and Pancakes!" Solution

3.6 F: Math Trade

```

1 class Part:
2     def __init__(self, name, has, wants):
3         self.name = name
4         self.has = has
5         self.wants = wants
6
7 people = []
8
9 n = int(input())

```



```

10
11 for i in range(n):
12     s = input().split()
13
14     x = Part(s[0], s[1], s[2])
15
16     people.append(x)
17
18 maxc = 0
19
20 for p in people:
21     c = 0
22     target = p
23     targets = []
24     while True:
25         inter = list(filter(lambda x: x.has == target.wants and x
26                             != target, people))
27
28         if len(inter) == 0:
29             break
30         else:
31             if inter[0] in targets:
32                 if c > maxc:
33                     maxc = c
34                     break
35
36             targets.append(inter[0])
37
38             c += 1
39             target = inter[0]
40
41 if maxc > 0:
42     print(maxc)
43 else:
44     print("No trades possible")

```

Listing 6: "Math Trade" Solution

3.7 G: Love Polygon

Missing content

3.8 H: Santa Klas

```

1 import math
2
3 s = input().split()
4
5 h = int(s[0])
6 a = int(s[1])
7
8 if a >= 0 and a <= 180:
9     print("safe")
10 else:
11     angle = a
12

```

```

13     if a > 180 and a < 270:
14         angle = 270 + (270 - angle)
15
16     angle = angle - 270
17
18     print(math.floor(h / (math.cos(math.radians(angle)))))

```

Listing 7: "Santa Klas" Solution

3.9 I: Triangle Ornaments

Missing content

3.10 J: Sacred Texts

```

1  from sys import stdin
2
3  def calcRuneValue(rune):
4      res = 0
5      for r in rune:
6          res += ord(r) - 32
7
8      return res
9
10 lines = stdin.readlines()
11
12 l1 = input().split()
13 rune = l1[0]
14 trans = l1[1]
15
16 letters = [chr(a) for a in range(ord("a"), ord("z") + 1)]
17
18 offset = letters.index(trans) - calcRuneValue(rune)
19
20 for i in range(8):
21     l = input()
22
23     if l == "":
24         break
25
26     res = ""
27
28     runes = l.split()
29
30     for r in runes:
31         if r == "0":
32             res += " "
33             continue
34         elif r == "<":
35             res += ","
36             continue
37         elif r == ">":
38             res += "."
39             continue
40
41     res += letters[(calcRuneValue(r) + offset) % len(letters)]

```

```

42
43     print(res)

```

Listing 8: "Sacred Texts" Solution

3.11 K: Spelling Bee

```

1  lets = input()
2  letters = [l for l in lets]
3  words = []
4
5  n = int(input())
6
7  for i in range(n):
8      word = input()
9      words.append(word)
10
11 for word in words:
12     if len(word) >= 4:
13         cond = True
14         for l in word:
15             if not l in letters:
16                 cond = False
17                 break
18         if not letters[0] in word:
19             cond = False
20
21     if cond:
22         print(word)

```

Listing 9: "Spelling Bee" Solution

3.12 L: Fifty Shades of Pink

```

1  n = int(input())
2
3  count = 0
4
5  for i in range(n):
6      col = input().lower()
7
8      if "pink" in col or "rose" in col:
9          count += 1
10
11 if count == 0:
12     print("I must watch Star Wars with my daughter")
13 else:
14     print(count)

```

Listing 10: "Fifty Shades of Pink" Solution

References

- [1] Kattis *Ranklist, scores and difficulties*, <https://open.kattis.com/help/ranklist>.
- [2] Kattis *Terms of Service*, <https://open.kattis.com/help/tos>