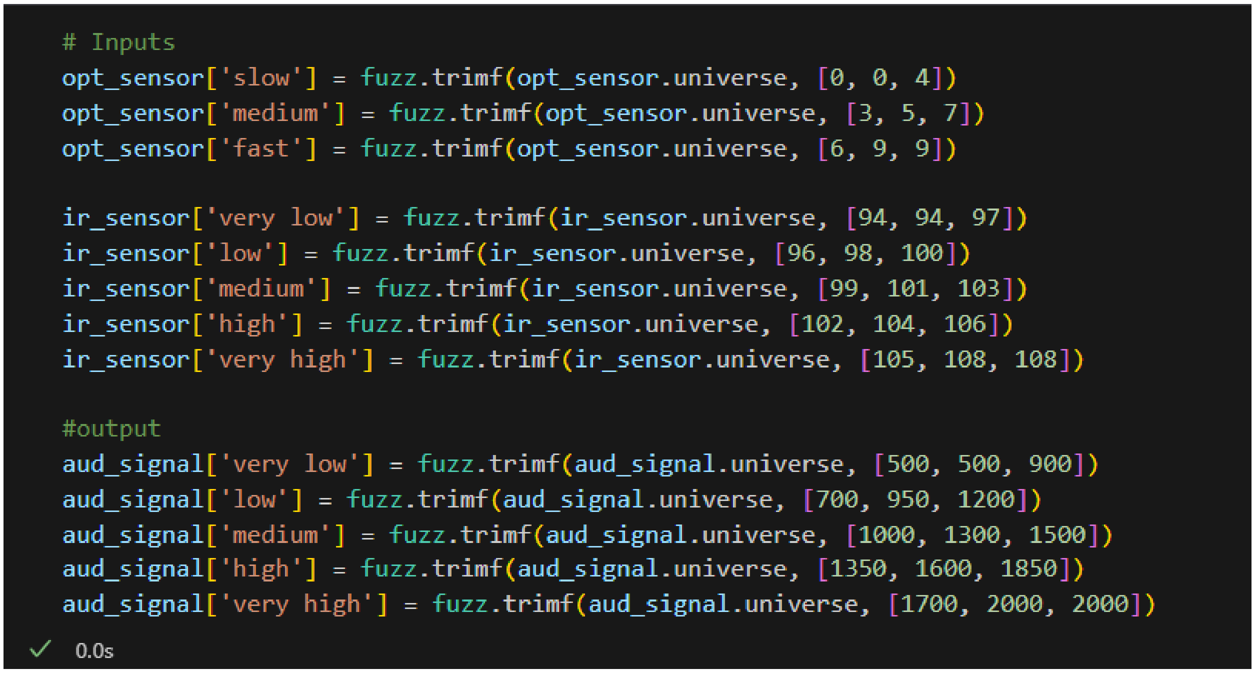
Name: Jayp S. Bazar Section: BSCS-2A

Personnel Detection System

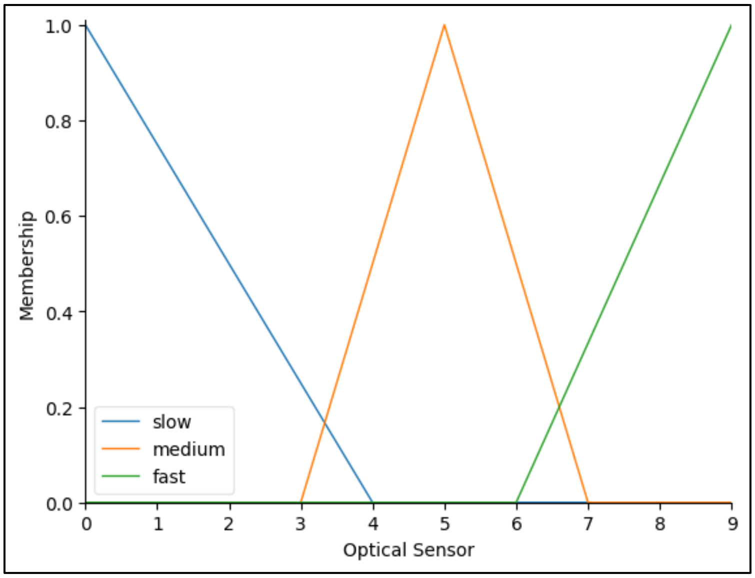
Membership Functions

Name	**Description**	**Values**		
Input:		Left	Center	Right
Optical Sensor	Slow	0	0	4
	Medium	3	5	7
	Fast	6	9	9
Infrared Sensor	Very Low	94	94	97
	Low	96	98	100
	Medium	99	101	103
	High	102	104	106
	Very High	105	108	108
Output:				
Audible Signal	Very Low	500	500	900
	Low	700	950	1200
	Medium	1000	1300	1500
	High	1350	1600	1850
	Very High	1700	2000	2000

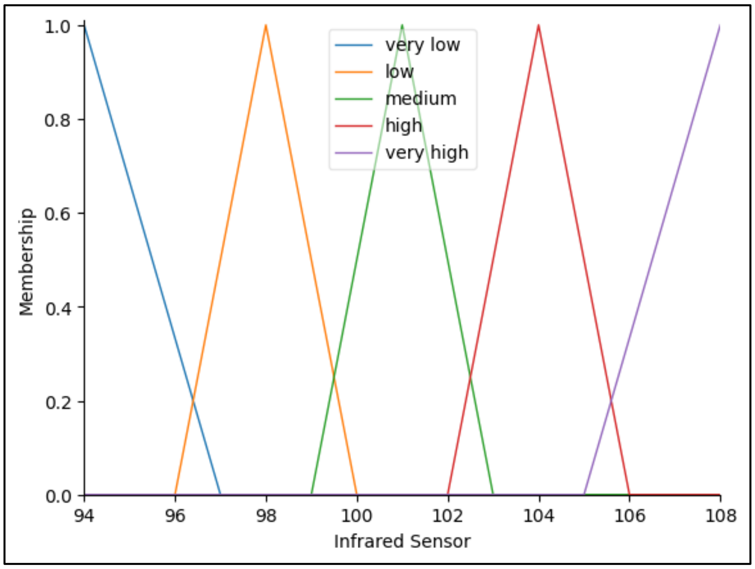


```
# Inputs
opt_sensor['slow'] = fuzz.trimf(opt_sensor.universe, [0, 0, 4])
opt_sensor['medium'] = fuzz.trimf(opt_sensor.universe, [3, 5, 7])
opt_sensor['fast'] = fuzz.trimf(opt_sensor.universe, [6, 9, 9])

ir_sensor['very low'] = fuzz.trimf(ir_sensor.universe, [94, 94, 97])
ir_sensor['low'] = fuzz.trimf(ir_sensor.universe, [96, 98, 100])
ir_sensor['medium'] = fuzz.trimf(ir_sensor.universe, [99, 101, 103])
ir_sensor['high'] = fuzz.trimf(ir_sensor.universe, [102, 104, 106])
ir_sensor['very high'] = fuzz.trimf(ir_sensor.universe, [105, 108, 108])

#output
aud_signal['very low'] = fuzz.trimf(aud_signal.universe, [500, 500, 900])
aud_signal['low'] = fuzz.trimf(aud_signal.universe, [700, 950, 1200])
aud_signal['medium'] = fuzz.trimf(aud_signal.universe, [1000, 1300, 1500])
aud_signal['high'] = fuzz.trimf(aud_signal.universe, [1350, 1600, 1850])
aud_signal['very high'] = fuzz.trimf(aud_signal.universe, [1700, 2000, 2000])
```

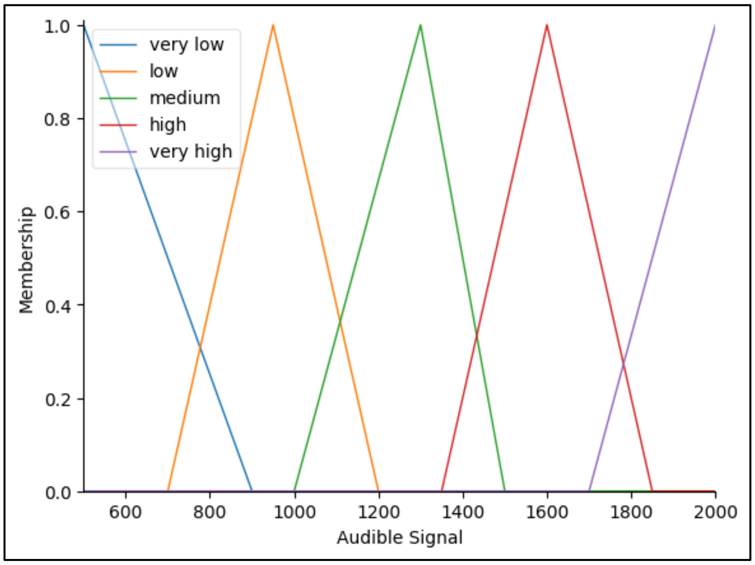
✓ 0.0s



Membership
1.0
0.8
0.6
0.4
0.2
0.0
slow
medium
fast
0
1
2
3
4
5
6
7
8
9
Optical Sensor



Membership
1.0
0.8
0.6
0.4
0.2
0.0
very low
low
medium
high
very high
94
96
98
100
102
104
106
108
Infrared Sensor



Membership
1.0
0.8
0.6
0.4
0.2
0.0
very low
low
medium
high
very high
600
800
1000
1200
1400
1600
1800
2000
Audible Signal

Rules

Rule 1 = If the infrared sensor is very low AND the optical sensor is slow, then the audio signal will be very low

Rule 2 = If the infrared sensor is very low AND the optical sensor is medium, then the audio signal will be low

Rule 3 = If the infrared sensor is very low AND the optical sensor is fast, then the audio signal will be medium

Rule 4 = If the infrared sensor is low AND the optical sensor is slow, then the audio signal will be low

Rule 5 = If the infrared sensor is low AND the optical sensor is medium, then the audio signal will be medium

Rule 6 = If the infrared sensor is low AND the optical sensor is fast, then the audio signal will be high

Rule 7 = If the infrared sensor is medium AND the optical sensor is slow, then the audio signal will be very low

Rule 8 = If the infrared sensor is medium AND the optical sensor is medium, then the audio signal will be low

Rule 9 = If the infrared sensor is medium AND the optical sensor is fast, then the audio signal will be medium

Rule 10 = If the infrared sensor is high AND the optical sensor is slow, then the audio signal will be low

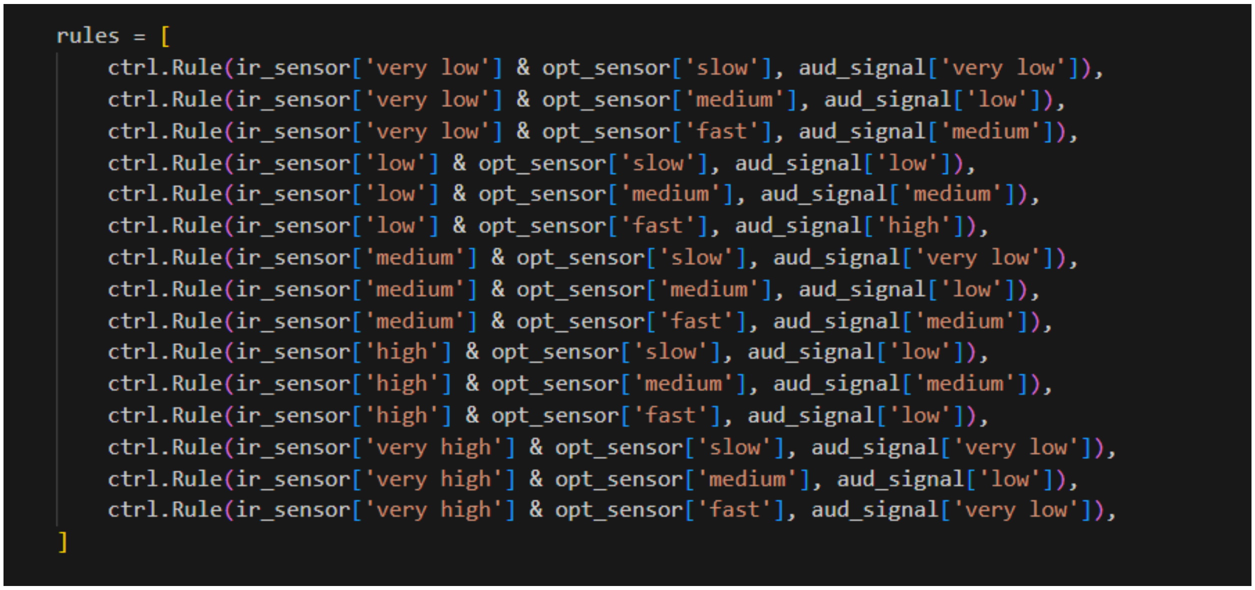
Rule 11 = If the infrared sensor is high AND the optical sensor is medium, then the audio signal will be medium

Rule 12 = If the infrared sensor is high AND the optical sensor is fast, then the audio signal will be low

Rule 13 = If the infrared sensor is very high AND the optical sensor is slow, then the audio signal will be very low

Rule 14 = If the infrared sensor is very high AND the optical sensor is medium, then the audio signal will be low

Rule 15 = If the infrared sensor is very high AND the optical sensor is fast, then the audio signal will be very low



```
rules = [
    ctrl.Rule(ir_sensor['very low'] & opt_sensor['slow'], aud_signal['very low']),
    ctrl.Rule(ir_sensor['very low'] & opt_sensor['medium'], aud_signal['low']),
    ctrl.Rule(ir_sensor['very low'] & opt_sensor['fast'], aud_signal['medium']),
    ctrl.Rule(ir_sensor['low'] & opt_sensor['slow'], aud_signal['low']),
    ctrl.Rule(ir_sensor['low'] & opt_sensor['medium'], aud_signal['medium']),
    ctrl.Rule(ir_sensor['low'] & opt_sensor['fast'], aud_signal['high']),
    ctrl.Rule(ir_sensor['medium'] & opt_sensor['slow'], aud_signal['very low']),
    ctrl.Rule(ir_sensor['medium'] & opt_sensor['medium'], aud_signal['low']),
    ctrl.Rule(ir_sensor['medium'] & opt_sensor['fast'], aud_signal['medium']),
    ctrl.Rule(ir_sensor['high'] & opt_sensor['slow'], aud_signal['low']),
    ctrl.Rule(ir_sensor['high'] & opt_sensor['medium'], aud_signal['medium']),
    ctrl.Rule(ir_sensor['high'] & opt_sensor['fast'], aud_signal['low']),
    ctrl.Rule(ir_sensor['very high'] & opt_sensor['slow'], aud_signal['very low']),
    ctrl.Rule(ir_sensor['very high'] & opt_sensor['medium'], aud_signal['low']),
    ctrl.Rule(ir_sensor['very high'] & opt_sensor['fast'], aud_signal['very low']),
]
```

2. In the given problem, can you adjust the membership values of the 3 variables to suit your needs? Yes or No. Explain your answer.

Yes, I can adjust the values of the membership functions to suit my needs. This is because the problem did not give specific values from the expert. Since I was assigned as both the programmer and the expert itself, I am the one to decide the values of the membership functions that will suit my needs.

3. Explain what difficulties can be encountered when this particular problem is solved using crisp logic.

Crisp logic is relying on definite values (true or false) which means it is not suitable to use in this kind of problems. It is not flexible enough compared to fuzzy logic when dealing with uncertainties. Using crisp logic to lacks the ambiguity of outputs and may lead to inaccurate conclusions.

4. Aside from the required outputs which is an audible sound for alarm, what other outputs do you think can be used here. Justify your answer.

There are multiple alternatives that can be used as a substitute for an audible sound alarm such as led lights or digital alerts like messages. For real-time alert, an alert using light can be a suitable replacement to the sound alarm. As for the digital alerts, it can be used in monitoring remotely and may be useful when the person in charge is not currently onsite.