Week: 03 Date: 9/11/2025

# 15-110 Recitation Week 3

## Reminders

- Check 2 due Monday, Sept. 15 at noon!
- HW 1 grades are out
- Check 1 and HW 1 Revisions due Tuesday, Sept. 16 at noon
- Recitation feedback form

## Overview

- Debugging
- Conditionals Practice
- Circuits and Gates

# Problems

### DEBUGGING

1. Rubber Duck Debugging 🦆

2. Printing and Experimenting

3. Thorough Tracing

**Catch the Error**

Read the following descriptions of what each function should return, along with the code itself. These are also in the starter file.

For each function, specify **the type of error** (syntax/runtime/logical), **error name,** and **what the error means.**

1) Return the average of three numbers:

```
 def computeIntAverage(x, y, z):
     sum = x + y + z
     count = 3
     return sum // count
print(computeIntAverage(5, “6”, 7))
```

| | |
|---|---|
| **Type** | |
| **Name** | |
| **What it means** | |

2) Return the intersection of two lines with slopes m1, m2 and y-intercepts b1 and b2, respectively:

```
def findLineIntersect(m1, b1, m2, b2):
      x = (b2 - b1) / (m1 - m1)
      y = m1 * x + b1
      print(‘y coord’, y)
      return x
```

| | |
|---|---|
| **Type** | |
| **Name** | |
| **What it means** | |

3) Return an appropriate statement depending on whether someone is leaving (i.e. leaving == True) or not:

```
def sayHelloOrGoodbye(leaving, name):
      if leaving == True:
            return “Hello ”+ name
      else:
            return “Goodbye, ”+ name+ “ and have a great day!
```

| | |
|---|---|
| **Type** | |
| **Name** | |
| **What it means** | |

4) Return the area of a rectangle with length and width x0 and x1:

```
def rectArea(x0, x1):
    area = x0 * x_1
    return area
```

| | |
|---|---|
| **Type** | |
| **Name** | |
| **What it means** | |

5) Return the distance between two points, (x1, y1, z1) and (x2, y2, z2), rounded up to the nearest integer (search up the distance formula if you need to!):

```
import math
def roundDistance(x1, y1, z1, x2, y2, z2):
      dX = x2 - x1**2
      dY = (y2 - y1) ** 2
      dZ = pow(2, (z1 - z2))
      dist = math.ceil(dX + dY + dZ) ** 0.5
      return dist
```

| | |
|---|---|
| **Type** | |
| **Name** | |
| **What it means** | |

## CONDITIONALS PRACTICE

In the heart of the fantastical town of Ticktockton lies the Enigmatic Clock Tower, which chimes magical melodies based on the time of day and the weather.

As the timekeeper of the Clock Tower, your responsibility is to decide the tower's chimes. The head timekeeper left you this python function to help:

Python

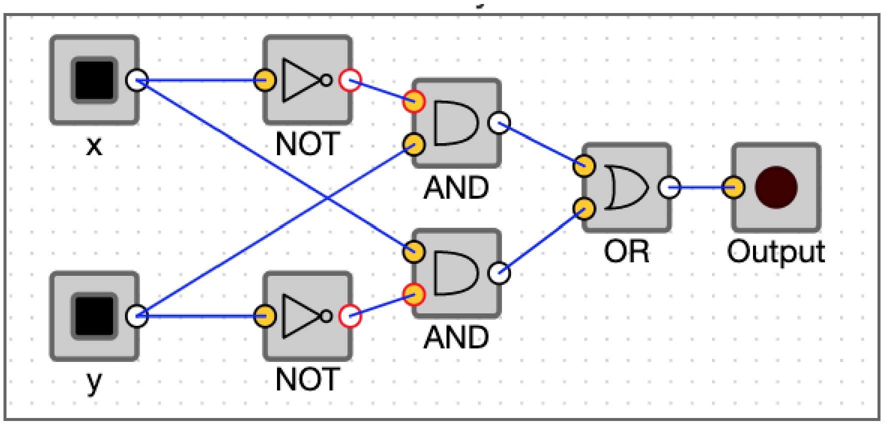
```python
def clockTower(time, weather):
    if time == "Morning":
        if weather == "Sunny":
            print("Chime of Cheerful Chipmunks")
        else:
            print("Melody of Morning Dew")
    elif time == "Afternoon":
        if weather == "Windy":
            print("Anthem of Airy Adventures")
        else:
            print("Ballad of Breezy Bluffs")
        print("With an encore Serenade of Starry Skies")
    elif weather == "Rainy":
        print("Raindrop Rhapsody")
    else:
        print("Twilight Tune")
```

Based on the specific combination of time of day + weather condition, this function will tell you (or print) the name of the song you should play. What will clockTower print for each function call below?

| **Function call** | **Printed value** |
|---|---|
| `clockTower(“Evening”, “Rainy”)` | |
| `clockTower(“Afternoon”, “Windy”)` | |
| `clockTower(“Morning”, “Rainy”)` | |
| `clockTower(“Night”, “Cloudy”)` | |

## CIRCUITS AND GATES

Write the equivalent Boolean expression demonstrated by the circuit:



Fill out the truth table that corresponds to the circuit and boolean expression above:

| x | y | output |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |