Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	[@KLWKS_BOT] THANOS

Experiment 1: Solving basic problems on McCulloch's pit neuron with python.

Aim/Objective: To implement and understand the functioning of McCulloch's pit neuron, demonstrating its binary logic operation.

Description: McCulloch's pit neuron takes binary inputs and produces a binary output. The neuron fires (output 1) if the sum of its inputs exceeds a certain threshold. In this experiment, we'll implement a simple McCulloch's pit neuron for logical AND and OR operations.

Pre-Requisites: Basic knowledge of binary logic, Python programming, and an understanding of basic neural network concepts.

Pre-Lab:

1. What is McCulloch's pit neuron?

McCulloch's pit neuron is a simplified mathematical model of a neuron that processes binary inputs using weights and a threshold to produce binary output (0 or 1).

2. What are the main parameters in McCulloch's pit neuron?

Main parameters: Inputs (x), Weights (w), Threshold (θ), Output (y).

3. How does McCulloch's pit neuron perform an AND operation?

AND operation: The neuron outputs 1 only if both inputs are 1. Example: If weights are 1 and threshold is 2, the sum must be 2 or more to output 1.

4. How does McCulloch's pit neuron perform an OR operation?

OR operation: The neuron outputs 1 if at least one input is 1. Example: If weights are 1 and threshold is 1, the sum must be 1 or more to output 1.

5. What is the significance of the weights and threshold in McCulloch's pit neuron's decision-making?

Significance of weights and threshold: Weights adjust input importance, and the threshold sets the limit for the neuron to activate (output 1).

In-Lab:

Program 1: Implement the basic logic gates AND & OR using McCulloch's Pit model.

- **Procedure/Program:**

```
x1 = [0, 0, 1, 1]
x2 = [0, 1, 0, 1]

w1, w2 = 1, 1

threshold_and = 2
threshold_or = 1

for i in range(4):
    net_and = x1[i] * w1 + x2[i] * w2
    y_and = 1 if net_and >= threshold_and else 0
    print(f"AND({x1[i]}, {x2[i]}) = {y_and}")

for i in range(4):
    net_or = x1[i] * w1 + x2[i] * w2
    y_or = 1 if net_or >= threshold_or else 0
    print(f"OR({x1[i]}, {x2[i]}) = {y_or}")
```

OUTPUT

```
AND(0, 0) = 0
AND(0, 1) = 0
AND(1, 0) = 0
AND(1, 1) = 1

OR(0, 0) = 0
OR(0, 1) = 1
OR(1, 0) = 1
OR(1, 1) = 1
```

- **Data and Results:**

```
DATA:
Binary input values are considered for logical gate operations.

RESULT:
Outputs of AND and OR gates are calculated and displayed.
```

- **Analysis and Inferences:**

```
ANALYSIS:
The logical behavior of AND and OR gates is verified.

INFERENCES:
McCulloch-Pitts model successfully implements basic logic gate functions.
```

Sample VIVA-VOCE Questions (In-Lab):

1. Provide a brief explanation of McCulloch's pit neuron concept?

McCulloch's pit neuron is a simple model that processes binary inputs using weights and a threshold to output either 0 or 1, mimicking a biological neuron for logical operations.

2. In your Python implementation of McCulloch's pit neuron, how did you determine the threshold and weights?

Threshold and weights in Python: The threshold is set based on the desired operation (e.g., 2 for AND), and weights are usually set to 1 for basic operations.

3. You implemented McCulloch's pit neuron for AND and OR operations. How did you set the parameters to perform these logical operations, and how does the neuron's output relate to the logic being implemented?

AND and OR operations: For AND, weights are 1, threshold is 2. For OR, weights are 1, threshold is 1. These settings match their truth tables.

4. If you wanted to extend its functionality to perform other logical operations, such as NOT, NAND, or XOR, how might you modify the neuron's parameters?

Other operations:

- **NOT**: Single input with threshold 0.
- **NAND**: Weights 1, threshold 2.
- **XOR**: Requires more complex modifications or multiple neurons.

Post-Lab:

Solve the given polynomial equation using TensorFlow $X^2 - 4X + 4 = 0$

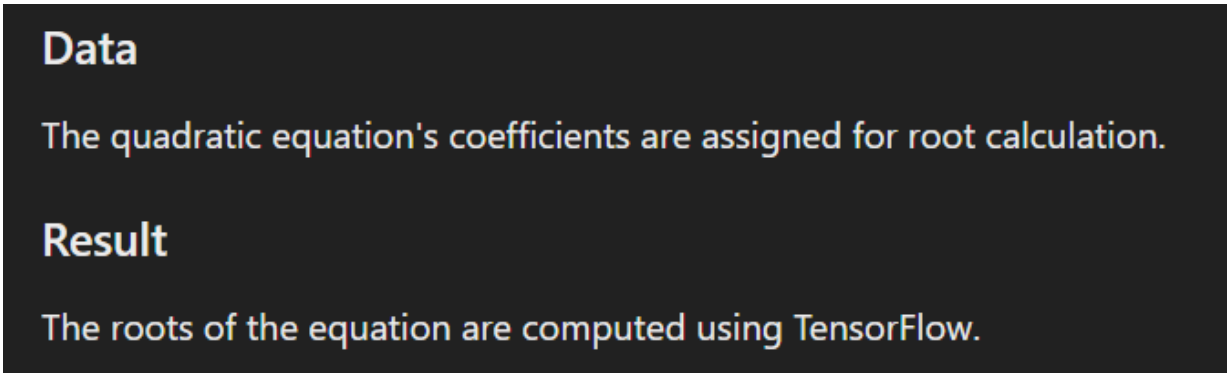
- **Procedure/Program:**

```
import tensorflow as tf

a = tf.constant(1.0)
b = tf.constant(-4.0)
c = tf.constant(4.0)

discriminant = b**2 - 4*a*c

sqrt_discriminant = tf.sqrt(discriminant)

root1 = (-b + sqrt_discriminant) / (2 * a)
root2 = (-b - sqrt_discriminant) / (2 * a)

print(f"Roots of the equation: {root1.numpy()}, {root2.numpy()}")
```

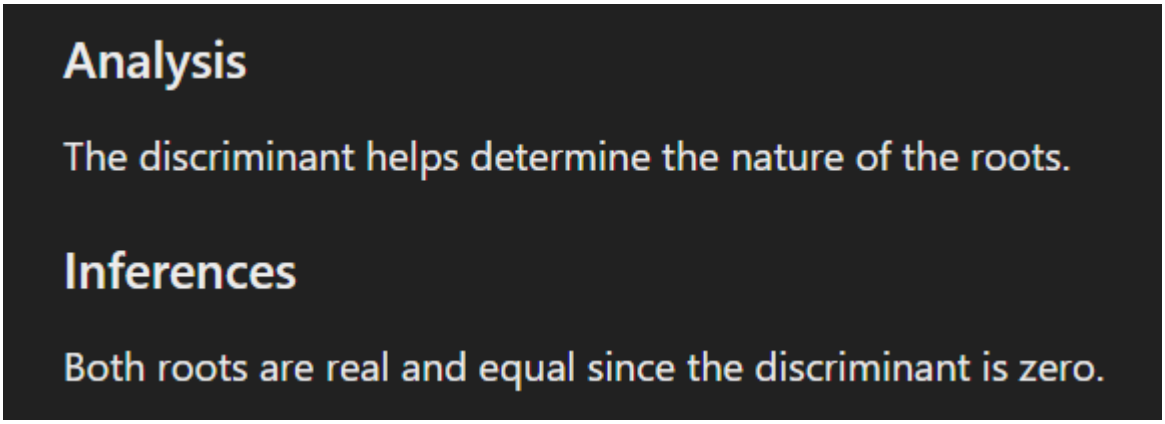
OUTPUT

```
Roots of the equation: 2.0, 2.0
```

- **Data and Results:**



- **Analysis and Inferences:**



Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.