PRACTICAL 2

A. AIM: Implement ANDNOT function using McCulloch-Pits neuron (use binary data representation

```
print("#######_____Methos_1_____########")
class McCullochPittsNeuron:
    def __init__(self, weights, threshold):
        self.weights = weights
        self.threshold = threshold
    def activate(self, inputs):
        if sum([x * w for x, w in zip(inputs, self.weights)]) >= self.threshold:
            return 1
        else:
            return 0
def ANDNOT(a, b):
    weights = [1, -1]
    threshold = 1
    neuron = McCullochPittsNeuron(weights, threshold)
    return neuron.activate([a, b])
print("ANDNOT(0, 0) =", ANDNOT(0, 0))
print("ANDNOT(0, 1) =", ANDNOT(0, 1))
print("ANDNOT(1, 0) =", ANDNOT(1, 0))
print("ANDNOT(1, 1) =", ANDNOT(1, 1))
print("#########_____Methos_2_____###########")
def mccullock_pitts_andnot(A,B):
    w1=1
    w2=-1
    threshold=0
    weighted_sum=w1*A+w2*B
    output=1 if weighted_sum>threshold else 0
    return output
input_A=int(input("Enter the value of A (0 or 1): "))
input_B=int(input("Enter the value of B (0 or 1): "))
if input_A in (0,1) and input_B in (0,1):
    result=mccullock_pitts_andnot(input_A,input_B)
    print(f"ANNOT({input_A},{input_B})={result}")
else:
    print("Invalid input.Please enter 0 or 1 for A and B.")
```

Output:

```
############Method_1############
ANDNOT(0, 0) = 0
ANDNOT(0, 1) = 0
ANDNOT(1, 0) = 1
ANDNOT(1, 1) = 0
############Method_2##############
Enter the value of A (0 or 1): 1
Enter the value of B (0 or 1): 0
ANNOT(1,0)=1
```

PRACTICAL 2

B. AIM: Generate XOR function using McCulloch-Pitts neural network.

```
import numpy as np
print("****** XOR CODE *********")
def sigmoid(x):
  return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
  return x * (1 - x)
try:
  A = int(input("Enter 1st Binary Input(0,1): "))
  B = int(input("Enter 2nd Binary Input (0,1): "))
  if A in [0, 1] and B in [0, 1]:
    # Create the XOR truth table
    x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
    y = np.array([[0], [1], [1], [0]])
    # Initialize the network parameters
    input_size = 2
    hidden_size = 2
    output_size = 1
    hidden_weight = np.random.uniform(size=(input_size, hidden_size))
    hidden_bias = np.random.uniform(size=(1, hidden_size))
    output_weights = np.random.uniform(size=(hidden_size, output_size))
    output_bias = np.random.uniform(size=(1, output_size))
    # Forward pass
    hidden_layer_input = np.dot(x, hidden_weight) + hidden_bias
    hidden_layer_output = sigmoid(hidden_layer_input)
    output_layer_input = np.dot(hidden_layer_output, output_weights)
     + output_bias
    output_layer_output = sigmoid(output_layer_input)
    # Print the output
    print(f"XOR({A}, {B}) = {(output_layer_output[0][0])}")
  else:
    print("Invalid Input. Please enter 0 or 1 for binary input.")
except:
  print("Invalid input.")
```

Output:

```
****** XOR CODE *********
Enter 1st Binary Input(0,1): 1
Enter 2nd Binary Input (0,1): 0
XOR(1, 0) = 0.7789986774398672
```