# Soft Computing Practical

1. a) Design a simple linear neural network model.

Code :-

```
x=float(input("Enter value of x:"))
w=float(input("Enter value of weight w:"))
b=float(input("Enter value of bias b:"))
net = int(w*x+b)
if(net<0):
        out=0
elif((net>=0)&(net<=1)):
        out =net
else:
        out=1
print("net=",net)
print("output=",out)
```

1. b) Calculate the output of neural net using both binary and bipolar sigmoidal function.

Code :-

```
print("the inputs")
inputs = [0.3,0.5,0.6]
print(inputs)
print("the weights")
weights = [0.2,0.1,-0.3]
print(weights)
print("The net input can be calculated as Yin = x1w1 + x2w2 + x3w3")
Yin = []
n=int(3)
for i in range(0, n):
   Yin.append(inputs[i]*weights[i])
print(round(sum(Yin),3))
```

2.a) Generate AND/NOT function using McCulloch-Pitts neural net.

Code :-

```
num_ip = int(4)
w1 = w2 =1
print("For the ", num_ip , " inputs calculate the net input using yin = x1w1 + x2w2 ")
x1 = [0,0,1,1]
x2 = [0,1,0,1]
print("x1 = ",x1)
print("x2 = ",x2)
n = x1 * w1
m = x2 * w2
Yin = []
for i in range(0, num_ip):
    Yin.append(n[i] + m[i])
print("Yin = ",Yin)
Yin = []
for i in range(0, num_ip):
    Yin.append(n[i] - m[i])
print("After assuming one weight as excitatory and the other as inhibitory Yin = ",Yin)
Y=[]
for i in range(0, num_ip):
    if(Yin[i]>=1):
        ele= 1
        Y.append(ele)
    if(Yin[i]<1):
        ele= 0
        Y.append(ele)
print("Y = ",Y)
```

2.b) Generate XOR function using McCulloch-Pitts neural net.

Code :-

```
# Simple McCulloch–Pitts XOR
for A in [0, 1]:
   for B in [0, 1]:
      # Hidden layer
      if (A + B) >= 1: # OR neuron
         H1 = 1
      else:
         H1 = 0
      if (A + B) >= 2: # AND neuron
         H2 = 1
      else:
         H2 = 0
      # Output neuron
      if (H1 - 2 * H2) >= 1:
         O = 1
      else:
         O = 0
      print(f"A={A}, B={B} -> XOR={O}")
```

3.a) write a program to implement Hebb's rule.

Code :-

```
import numpy as np
x1=np.array([1,1,1,-1,1,-1,1,1,1])
x2=np.array([1,1,1,1,-1,1,1,1,1])
b=0
y=np.array([1,-1])
wtnew=np.zeros((9,))
print("First input with target =1")
for i in range(0,9):
```

```
    wtnew[i]=wtnew[i]+x1[i]*y[0]
b=b+y[0]
print("new wt =", wtnew)
print("Bias value",b)
print("Second input with target =-1")
for i in range(0,9):
    wtnew[i]=wtnew[i]+x2[i]*y[1]
b=b+y[1]
print("new wt =", wtnew)
print("Bias value",b)
```

3.b) Write a program to implement of delta rule.

Code :-

```
#supervised learning
import numpy as np
x=np.array([1,1,1])
weights=np.array([1,1,1])
desired=np.array([2,3,4])
actual=np.zeros((3,))
a=1
print("Learning Rate : 1")
actual=x*weights
print("actual",actual)
print("desired",desired)
while True:
    if np.array_equal(desired,actual):
        break #no change
    else:
        for i in range(0,3):
            weights[i]=weights[i]+a*(desired[i]-actual[i])
        actual=x*weights
```

```
        print("weights",weights)
        print("actual",actual)
        print("desired",desired)
print("*"*30)
print("Final output")
print("Corrected weights",weights)
print("actual",actual)
print("desired",desired)
```