Q1. Perform "Step", "Sigmoid", "Relu" activation functions

```
import numpy as np

def art_neuron(w, x, b, activation='step'):
  z = np.dot(w, x) + b
  if activation == 'step':
    return 1 if z >= 0 else 0
  elif activation == 'sigmoid':
    return 1 / (1 + np.exp(-z))
  elif activation == 'relu':
    return np.maximum(0, z)

x = np.array([0.8, 0.3])
w = np.array([0.2, 0.1])
b = 0.5

def sigmoid_direct(z):
    return 1 / (1 + np.exp(-z))

def step_direct(z):
    return np.where(z >= 0, 1, 0)

def relu_direct(z):
    return np.maximum(0, z)

print("\n\nSigmoid Output : ", art_neuron(w, x, b, 'sigmoid'))
print("Step Output : ", art_neuron(w, x, b, 'step'))
print("Relu Output : ", art_neuron(w, x, b, 'relu'))
```

```


Sigmoid Output :  0.6659669267518202
Step Output :  1
Relu Output :  0.6900000000000001
```

Q2. Mc Cullouch pitts neuron for logic gates AND & OR

```
#AND
def mp_and(x1, x2):
    w1, w2 = 1, 1
    theta = 2
    z = x1*w1 + x2*w2
    return 1 if z >= theta else 0

print(" AND GATE using Mc Culloch Pitts Neuron\n")
for x1 in [0, 1]:
    for x2 in [0, 1]:
        print({x1}, {x2},"-", mp_and(x1, x2))
```

```
 AND GATE using Mc Culloch Pitts Neuron

{0} {0} - 0
{0} {1} - 0
{1} {0} - 0
{1} {1} - 1
```

```
#OR
def mp_or(x1, x2):
    w1, w2 = 1, 1
    theta = 1
    z = x1*w1 + x2*w2
    return 1 if z >= theta else 0

print(" OR GATE using Mc Culloch Pitts Neuron\n")
for x1 in [0, 1]:
    for x2 in [0, 1]:
        print({x1}, {x2},"-", mp_or(x1, x2))
```

```
 OR GATE using Mc Culloch Pitts Neuron

{0} {0} - 0
{0} {1} - 1
{1} {0} - 1
{1} {1} - 1
```

Q3. Perceptron through learning algorithm

```
def perceptron(x, w, b):
    z = np.dot(w, x) + b
    return 1 if z >= 0 else 0
```