In [1]:

```python
import numpy as np
```

In [2]:

```python
def unitStep(v):
    if v >= 0:
        return 1
    else:
        return 0
```

In [3]:

```python
def perceptronModel(x, w, b):
    v = np.dot(w, x) + b
    y = unitStep(v)
    return y
```

In [4]:

```python
def NOT_logicFunction(x):
    wNOT = -1
    bNOT = 0.5
    return perceptronModel(x, wNOT, bNOT)
```

In [5]:

```python
def AND_logicFunction(x):
    w = np.array([1, 1])
    bAND = -1.5
    return perceptronModel(x, w, bAND)
```

In [6]:

```python
def OR_logicFunction(x):
    w = np.array([1, 1])
    bOR = -0.5
    return perceptronModel(x, w, bOR)
```

In [7]:

```python
def XOR_logicFunction(x):
    y1=AND_logicFunction(x)
    y2=OR_logicFunction(x)
    y3=NOT_logicFunction(y1)
    final_x=np.array([y2,y3])
    final_output=AND_logicFunction(final_x)
    return final_output
```

In [8]:

```python
test1=np.array([0,0])
test2=np.array([0,1])
test3=np.array([1,0])
test4=np.array([1,1])
```

In [9]:

```
print("XOR({},{})={}".format(0,0,XOR_logicFunction(test1)))
print("XOR({},{})={}".format(0,1,XOR_logicFunction(test2)))
print("XOR({},{})={}".format(1,0,XOR_logicFunction(test3)))
print("XOR({},{})={}".format(1,1,XOR_logicFunction(test4)))
```

```
XOR(0,0)=0
XOR(0,1)=1
XOR(1,0)=1
XOR(1,1)=0
```

In [ ]: