In [1]:
```
class McCullochPittsNeuron:
    def __init__(self,weights,threshold):
        self.weights=weights
        self.threshold=threshold
    def activate(self,inputs):
        weighted_sum=sum([inputs[i]*self.weights[i]
        for i in range(len(inputs))])
        return 1 if weighted_sum>=self.threshold  else 0
and_not_weights=[1,-1]
and_not_threshold=1
and_not_neuron=McCullochPittsNeuron(and_not_weights,and_not_threshold)
input_values_and_not=[(0,0),(0,1),(1,0),(1,1)]
print("And_not function:")
for inputs in input_values_and_not:
    output=and_not_neuron.activate(inputs)
    print(f"input:{inputs},output:{output}")
nand_weights=[-1,-1]
nand_threshold=-1
nand_neuron=McCullochPittsNeuron(nand_weights,nand_threshold)
or_weights=[1,1]
or_threshold=1
or_neuron=McCullochPittsNeuron(or_weights,or_threshold)
xor_weights=[1,1]
xor_threshold=2
xor_neuron=McCullochPittsNeuron(xor_weights,xor_threshold)
input_values_xor=[(0,0),(0,1),(1,0),(1,1)]
print("xor function:")
for inputs in input_values_xor:
    output=xor_neuron.activate((nand_neuron.activate(inputs),or_neuron.activate(inputs)))
    print(f"input:{inputs},output:{output}")
```

```
And_not function:
input:(0, 0),output:0
input:(0, 1),output:0
input:(1, 0),output:1
input:(1, 1),output:0
xor function:
```

```
input:(0, 0),output:0
input:(0, 1),output:1
input:(1, 0),output:1
input:(1, 1),output:0
```

In [ ]: