Importing libraries

In [1]:

```
from random import *
```

Logical gates implementation

Every function has been implemented as a MCP Neuron

In [2]:

```
def and_gate(arg1, arg2):
    wt = [1, 1]
    threshold = 2

    if wt[0]*arg1 + wt[1]*arg2 >= threshold: return 1
    else: return 0


def not_gate(arg1):
    wt = [-1]
    threshold = 0

    if wt[0]*arg1 >= threshold: return 1
    else: return 0


def or_gate(arg1, arg2):
    wt = [1, 1]
    threshold = 1

    if wt[0] * arg1 + wt[1] * arg2 >= threshold: return 1
    else: return 0


def xor_gate(arg1, arg2):
    return or_gate(and_gate(arg1, not_gate(arg2)), and_gate(not_gate(arg1
), arg2))


def half_adder(arg1, arg2):
    add = xor_gate(arg1, arg2)
    carry = and_gate(arg1, arg2)
    return add, carry
```

Applying the inputs to the MCP Neurons

Inputs are standard logic gate inputs

In [3]:

```
for ip in [[0, 0], [0, 1], [1, 0], [1, 1]]:
    print('{0} AND {1} = {2}'.format(ip[0], ip[1], and_gate(ip[0], ip[1
])))
    print('{0} OR {1} = {2}'.format(ip[0], ip[1], or_gate(ip[0], ip[1])))
    ans = half_adder(ip[0], ip[1])
    print('{0} ADD {1} : SUM = {2} CARRY = {3}'.format(ip[0], ip[1], ans[
0], ans[1]))
    print('-'*30)

for ip in [randrange(-10, 5) for _ in range(10)][::2][:3]:
    print('NOT {0} = {1}'.format(ip, not_gate(ip)))
```

```
0 AND 0 = 0
0 OR 0 = 0
0 ADD 0 : SUM = 0 CARRY = 0
------------------------------
0 AND 1 = 0
0 OR 1 = 1
0 ADD 1 : SUM = 1 CARRY = 0
------------------------------
1 AND 0 = 0
1 OR 0 = 1
1 ADD 0 : SUM = 1 CARRY = 0
------------------------------
1 AND 1 = 1
1 OR 1 = 1
1 ADD 1 : SUM = 0 CARRY = 1
------------------------------
NOT 3 = 0
NOT 2 = 0
NOT -4 = 1
```