**Name : Darshil Maru**

**Roll No : 20bce514**

**Practical 4 : Byzantine Problem**

The Byzantine Generals Problem is a game theory problem, which describes the difficulty decentralized parties have in arriving at consensus without relying on a trusted central party. In a network where no member can verify the identity of other members, how can members collectively agree on a certain truth?

In [15]:

```
from collections import Counter
```

In [16]:

```
class General:
    def __init__(self, id, is_traitor=False):
        self.id = id
        self.other_generals = []
        self.orders = []
        self.is_traitor = is_traitor
    def __call__(self, m, order):
        self.byzantine_algorithm(commander=self, m=m, order=order)
    def _next_order(self,is_traitor,order,i):
        if is_traitor:
            if i % 2 == 0:
                return "Attack" if order == "Retreat" else "Retreat"
        return order
    def byzantine_algorithm(self, commander, m, order):
        if m<0:
            self.orders.append(order)
        elif m ==0:
            for i, l in enumerate(self.other_generals):
                l.byzantine_algorithm(commander=self, m=(m-1), order=self._next_order(self.
        else:
            for i, l in enumerate(self.other_generals):
                if i is not self and l is not commander:
                    l.byzantine_algorithm(commander=self, m=(m-1), order=self._next_order(s
    @property
    def decision(self):
        c = Counter(self.orders)
        return (c.most_common())
```

In [17]:

```
def init_generals(generals_spec):
    generals =[]
    for i, spec in enumerate(generals_spec):
        general = General(i)
        if spec == "l":
            pass
        elif spec == "t":
            general.is_traitor = True
        else:
            print("Incorrect input")
            exit(1)
        generals.append(general)
    for general in generals :
        general.other_generals = generals
    return generals
```

In [18]:

```
def print_decision(generals):
    for i, l in enumerate(generals):
        print("General {}: {}".format(i, l.decision))
```

In [19]:

```
m = 0
g = "l, l, l"
o = "Attack"
```

In [20]:

```
generals_spec = [x.strip() for x in g.split(',')]
generals = init_generals(generals_spec=generals_spec)
generals[0](m=m, order=o)
print_decision(generals)
```

```
General 0: [('Attack', 1)]
General 1: [('Attack', 1)]
General 2: [('Attack', 1)]
```

In [21]:

```
m = 2
g = "l, l, t, t, l, l"
o = "Attack"
```

In [22]:

```
generals_spec = [x.strip() for x in g.split(',')]
generals = init_generals(generals_spec=generals_spec)
generals[0](m=m, order=o)
print_decision(generals)
```

```
General 0: [('Attack', 15), ('Retreat', 10)]
General 1: [('Attack', 21), ('Retreat', 4)]
General 2: [('Attack', 15), ('Retreat', 10)]
General 3: [('Attack', 21), ('Retreat', 4)]
General 4: [('Attack', 15), ('Retreat', 10)]
General 5: [('Attack', 21), ('Retreat', 4)]
```