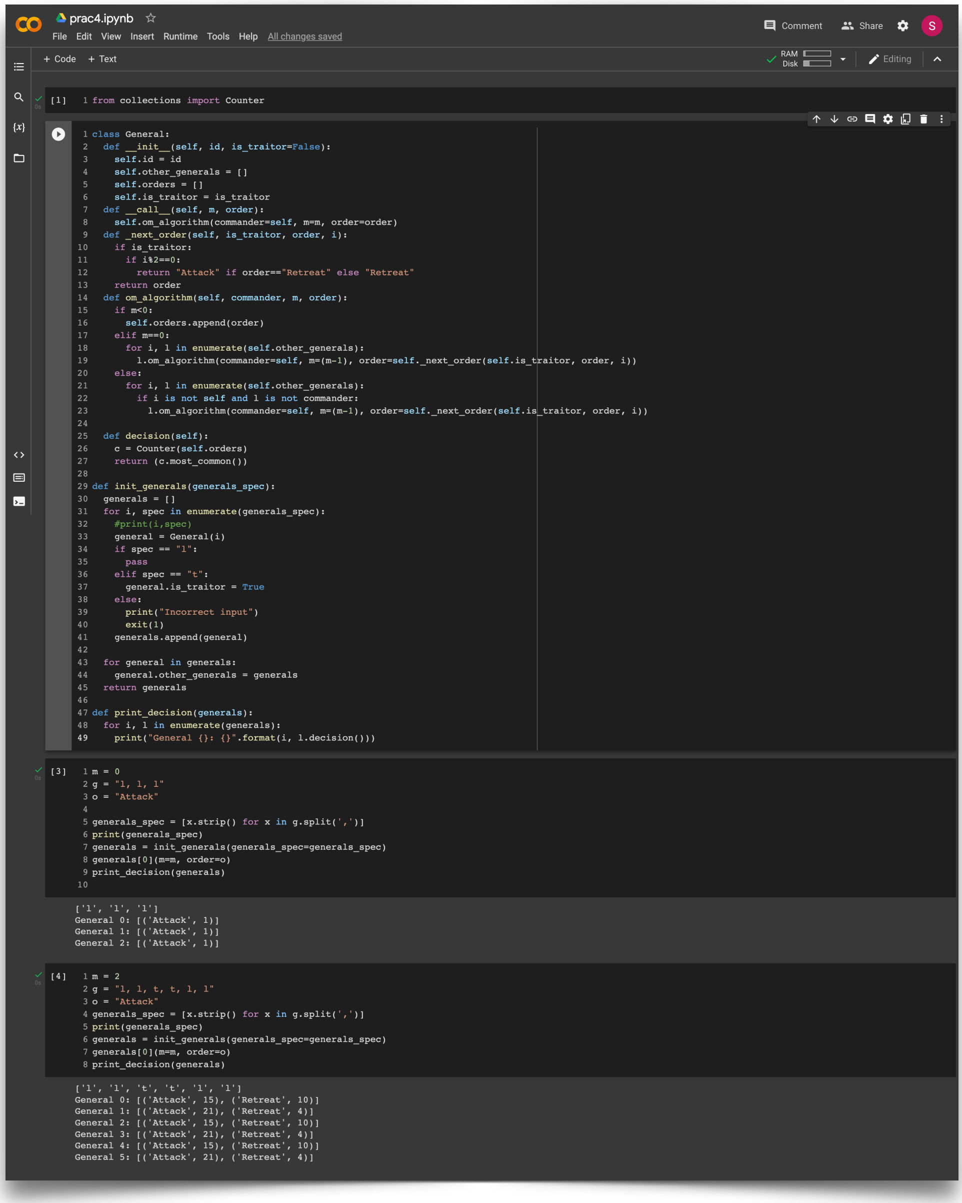
Aayush Shah
19BCE245
22 October 2022

BlockChain Technology

Practical 4

Implementing Byzantine Fault Tolerance

- **Code :**



[1]
```python
from collections import Counter
```

```python
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.om_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 om_algorithm(self, commander, m, order):
    if m<0:
      self.orders.append(order)
    elif m==0:
      for i, l in enumerate(self.other_generals):
        l.om_algorithm(commander=self, m=(m-1), order=self._next_order(self.is_traitor, order, i))
    else:
      for i, l in enumerate(self.other_generals):
        if i is not self and l is not commander:
          l.om_algorithm(commander=self, m=(m-1), order=self._next_order(self.is_traitor, order, i))

  def decision(self):
    c = Counter(self.orders)
    return (c.most_common())

def init_generals(generals_spec):
  generals = []
  for i, spec in enumerate(generals_spec):
    #print(i,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

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

[3]
```python
m = 0
g = "l, l, l"
o = "Attack"

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

```

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

[4]
```python
m = 2
g = "l, l, t, t, l, l"
o = "Attack"
generals_spec = [x.strip() for x in g.split(',')]
print(generals_spec)
generals = init_generals(generals_spec=generals_spec)
generals[0](m=m, order=o)
print_decision(generals)
```

```
['l', 'l', 't', 't', 'l', 'l']
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)]
```

Conclusion

In this practical, I understood the working of practical byzantine fault tolerance in Blockchain systems, the math behind this algorithm.

.