Jonathan Woodhouse

```
# package to handle thread-base parallelism
import threading
#package to handle time related functions
import time

print ("imported some packages")
```

```
imported some packages
```

```
# represents a distibuted database system
class DatabaseNode:
  # initialization method
  def __init__(self, node_id):
    # unique identifier for each node
    self.node_id = node_id
    # data stored locally within the node
    self.data = {}
    # list of replica nodes
    self.replica_nodes = []

# simulates a write operation on the database node
def write_data(self, key, value):
  print(f"Node {self.node_id}: Write Operation - Key: {key}, Value: {value}")
  self.data [key] = value
  # iterates over each replica node to replicate the write operations
  for replica_node in self.replica_nodes:
    replica_node.recive_replication(key,value)

# recieve replicated data from other nodes
def recieve_replication(self, key, value):
    print(f"Node {self.node_id}: Replication - Key: {key}, Value: {self.data.get(key, 'Not found')}")
    return self.data.get(key,None)

print("created the node that represents the distributed database system")
```

```
created the node that represents the distributed database system
```

```
# simulates a continuous stream of write operations on a database node
def simulate_writes(node):
  # used to generate unique keys for write operation
  i - 0
  # continuous loop
  while True:
    node.write_data(f" k - {i}", f" v - {i}")
    # ensure unique key-value pair
    i += 1
    # pause execution for 2 seconds before the next iteration
    time.sleep(2)

  print("defined the methods to handle simulating a continuous stream of write operations")
```

Generate 10 random numbers using numpy Close

```
# create two node instances
node1 = DatabaseNode(1)
node2 = DatabaseNode(2)

#set up replication between the two nodes
node1.replica_nodes.append(node2)
node2.replica_nodes.append(node1)

print("initialized the node instances and setup node replication")
```

```
initialized the node instances and setup node replication
```

```
# start write operations for nodel in a seperate thread
threading.Thread(target=simulate_writes, args=(nodel,)).start()
```

```
# initiates a read operation node1
node1.read_data("key0")
# pause 5 seconds to all write operations to be replicated between the nodes before reading again
time.sleep(5)
# performs a similar read operation on node2, allow for replication of write operations between the nodes
node2.read_data("key0")
```