Deadlock simulation

In [ ]:
```
import time
from threading import Thread, RLock
```

In [ ]:
```
class Shared:
    def __init__(self):
        self.lock = RLock()

    def test1(self, arg):
        with self.lock:
            print("test 1 begin")
            time.sleep(1)
            arg.test2(self)
            print("test 1 end")

    def test2(self, arg):
        with self.lock:
            print("test 2 begin")
            time.sleep(1)
            arg.test1(self)
            print("test 2 end")


class Thread1(Thread):
    def __init__(self,s1,s2):
        super().__init__()
        self.s1 = s1
        self.s2 = s2

    def run(self):
        self.s1.test1(self.s2)

class Thread2(Thread):
    def __init__(self,s1,s2):
        super().__init__()
        self.s1 = s1
        self.s2 = s2

    def run(self):
        self.s2.test2(self.s1)
```

In [ ]:
```
s1 = Shared()
s2 = Shared()

t1 = Thread1(s1,s2)
t1.start()

t2 = Thread2(s1,s2)
t2.start()

time.sleep(2)
```

```
test 1 begin
test 2 begin
```

Deadlock in centrailzed system

In [ ]:
```
def MyInput(smg):
    print(smg, end ="")
    ans = input()
    print(ans)
    return ans

p = int(MyInput("Enter the number of processes: "))
r = int(MyInput("Enter the number of resources: "))
arr = [0] * p
for i in range(p):
    arr[i]=int(MyInput(f"Process{i+1}usingresource:"))
count = 0
for i in range(0, len(arr)):
    for j in range(i + 1, len(arr)):
        if arr[i] == arr[j]:
            count += 1
if count > 0:
    print("Deadlock Present")
else:
     print("Deadlock Absent")
```

```
Enter the number of processes: 4
Enter the number of resources: 3
Process1usingresource:1
Process2usingresource:2
Process3usingresource:1
Process4usingresource:2
Deadlock Present
```

Deadlock in distributed system

In [ ]:
```
p = int(MyInput("Enter number of process: "))
r = int(MyInput("Enter number of resource: "))
resource = [0] * p
for i in range(p):
    resource[i] = int(MyInput(f"Process {i + 1} using resource: "))
flag = 0
for i in resource:
    flag += 1
    z = resource.count(i)
    if z > 1:
        print(f"Process {flag} has failed due to dead lock caused")
    else:
        print(f"Process {flag} is still working as we have followed Distributed approa
```

```
Enter number of process: 4
Enter number of resource: 3
Process 1 using resource: 1
Process 2 using resource: 2
Process 3 using resource: 1
Process 4 using resource: 2
Process 1 has failed due to dead lock caused
Process 2 has failed due to dead lock caused
Process 3 has failed due to dead lock caused
Process 4 has failed due to dead lock caused
```

In [ ]:
```
```