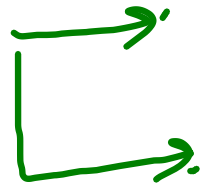
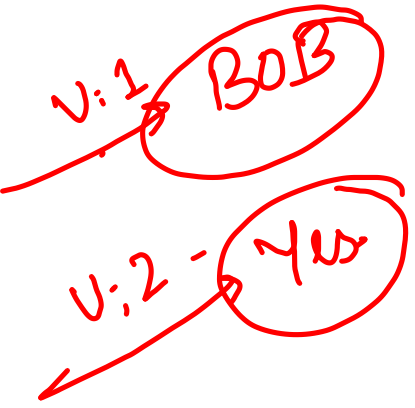
Resource Management



1. Concurrency
2. Adv & drawbacks
3. Synchronization
4. What are the cond^n to achieve Synchronization
5. Softw. 6. H/W.

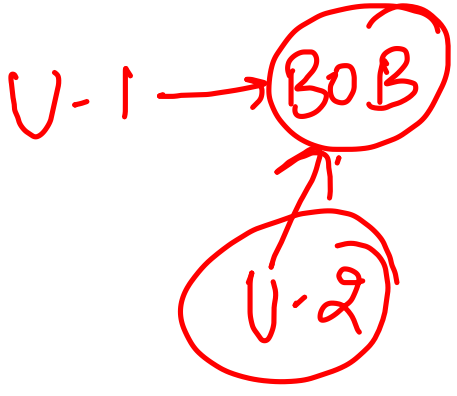
1) P.M.
2) M.M
3) R.M
4) F.M

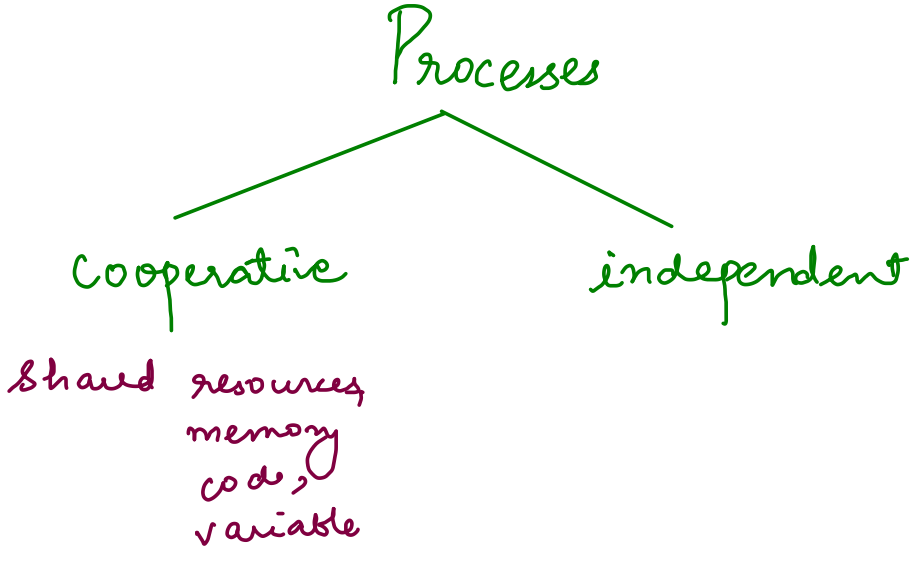


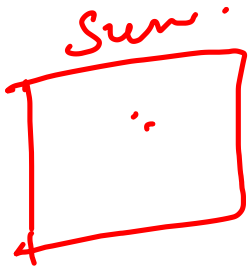
Concurrency

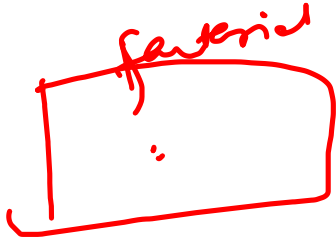
thread

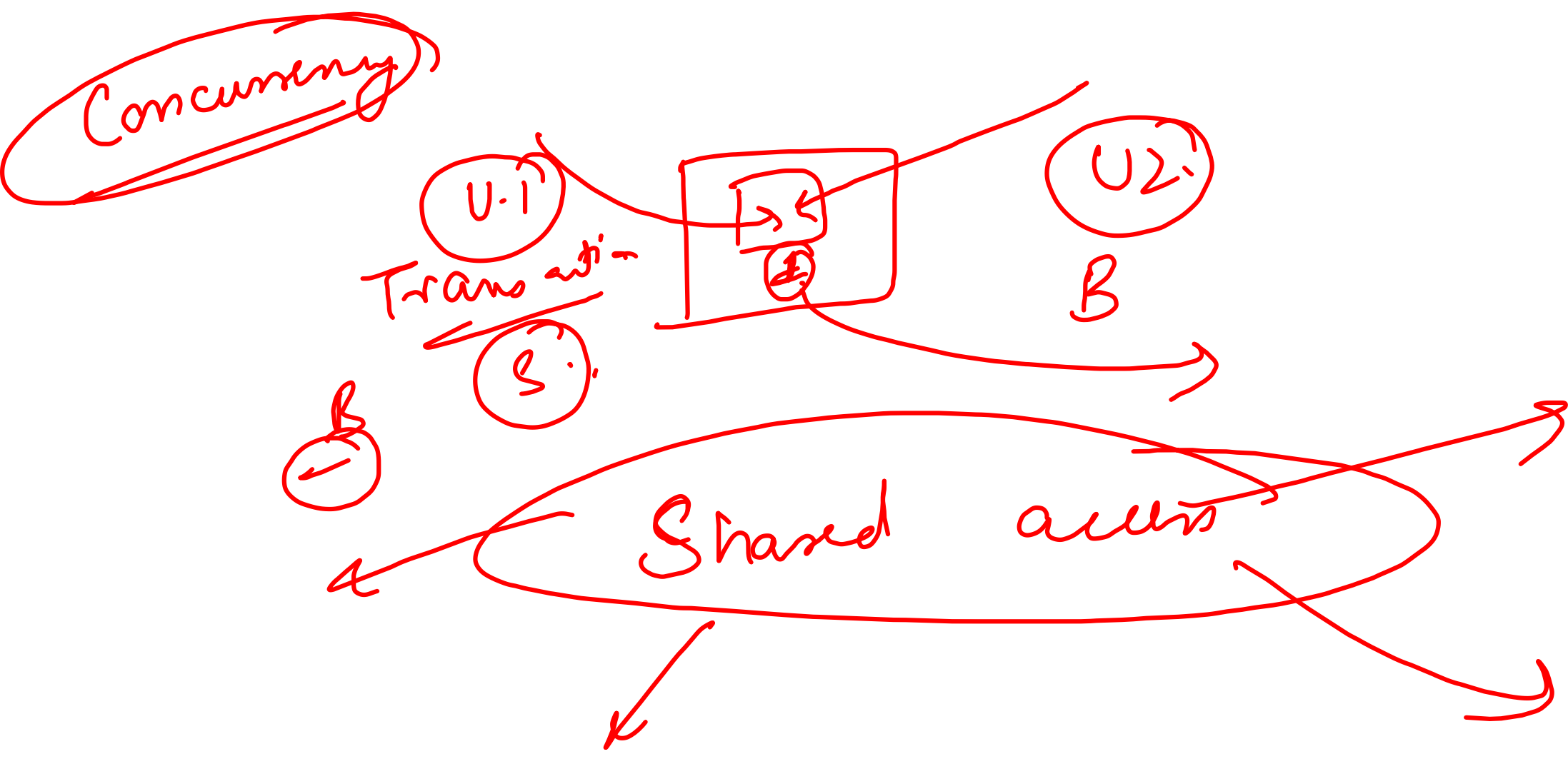
State in which a process exist simultaneously with another process then it is concurrent in nature.

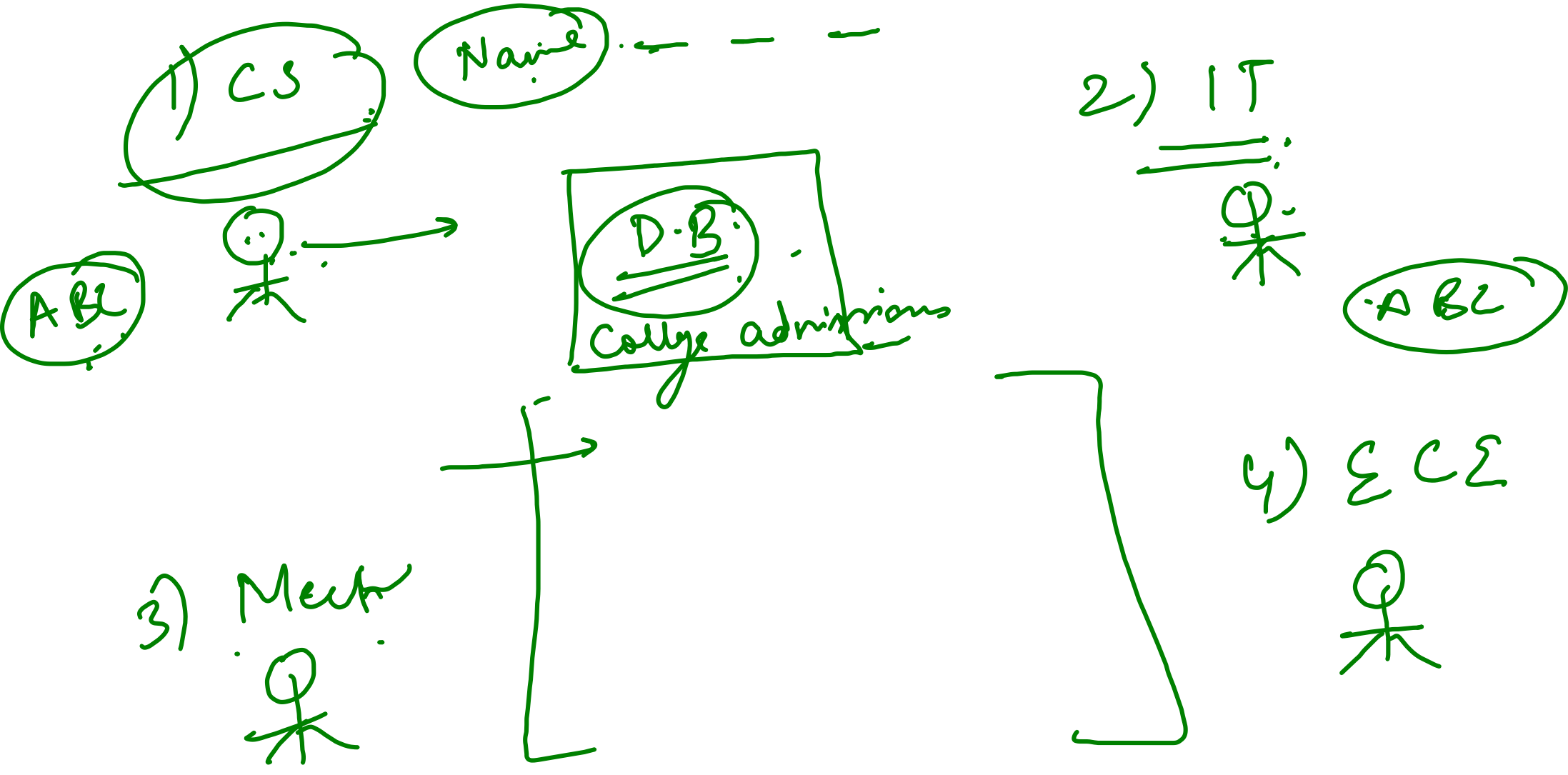


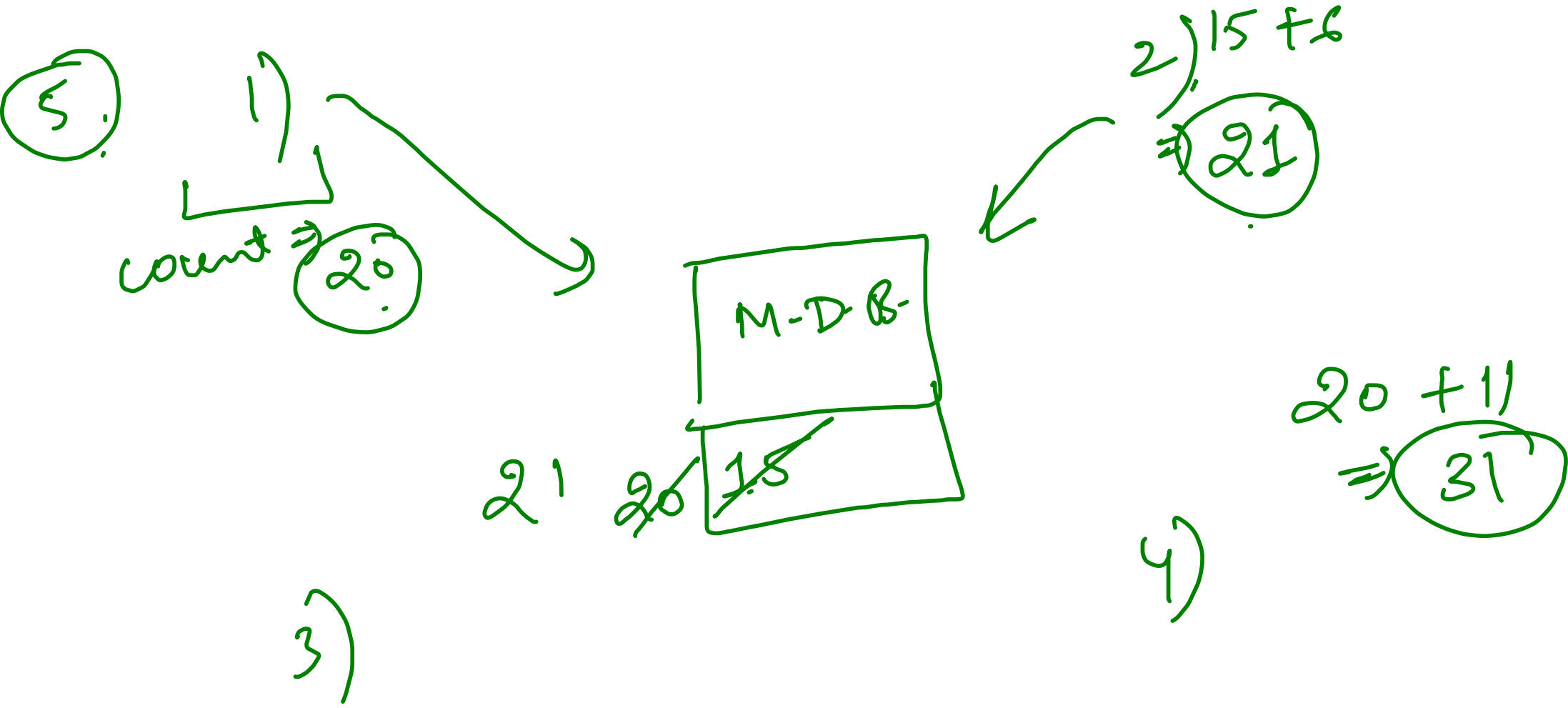







Concurrency
U.1
Transaction
S
U2
B
B
Shared access

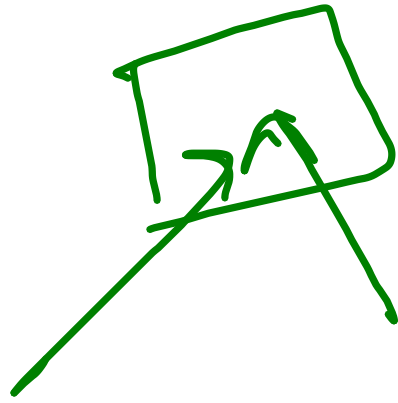

1) CS
Name
2) IT
D.B
college admissions
ABC
ABC
3) Mech
4) ECE


5
1)
count ⇒ 20
2) 15 + 6
⇒ 21
M-D-B-
21
20
15
20 + 11
⇒ 31
3)
4)

Advantages :-

1) Better resource utilization.

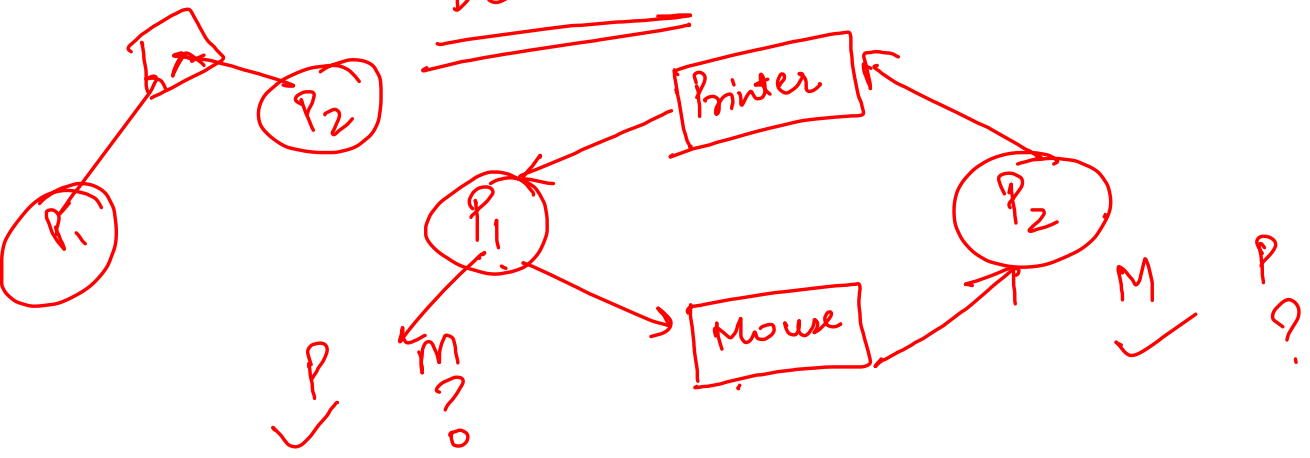
2) Better performance.

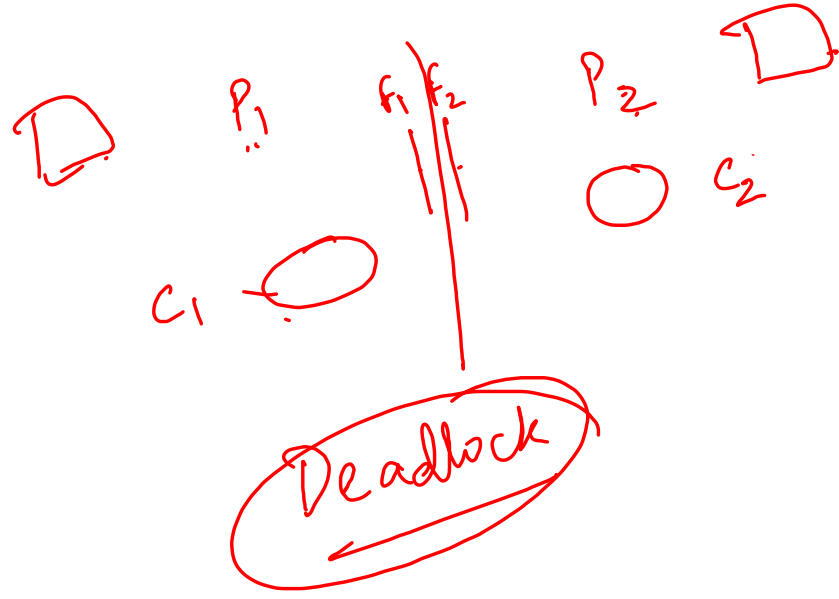


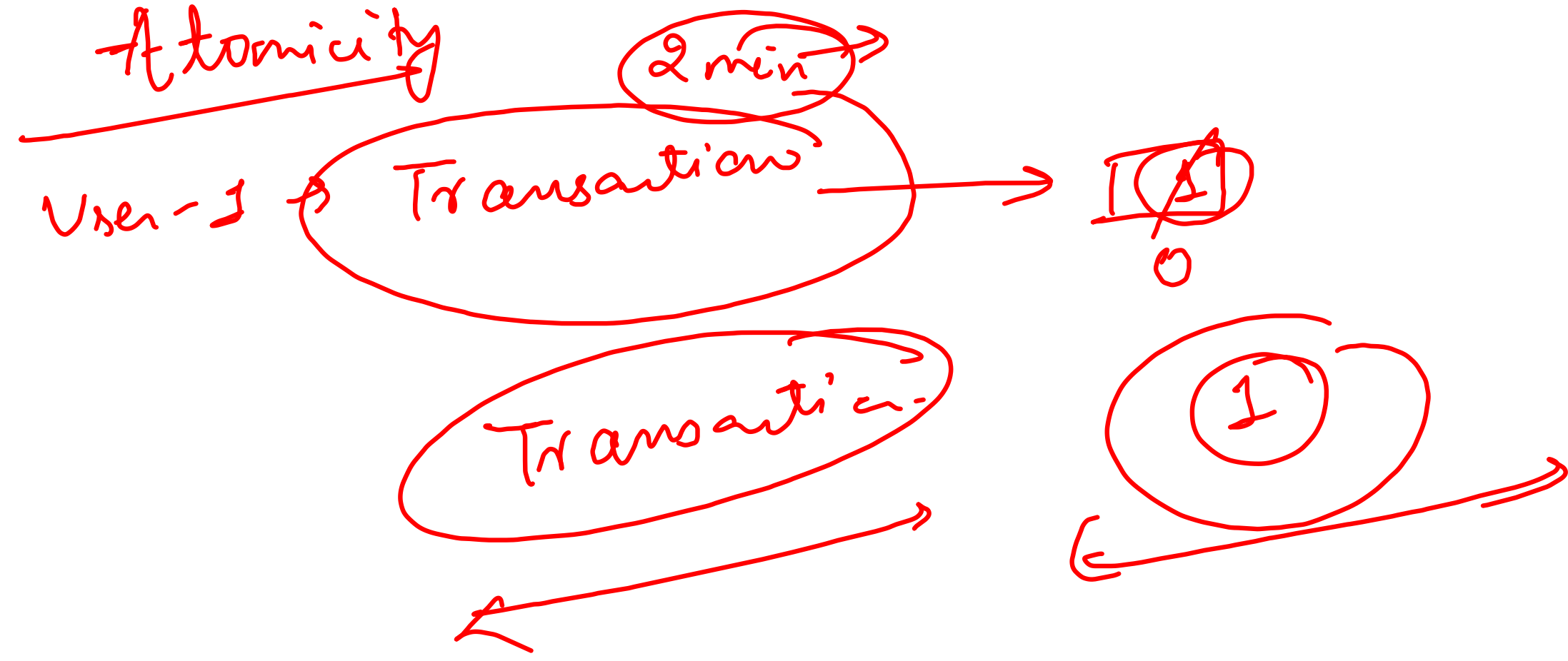
Issues :-

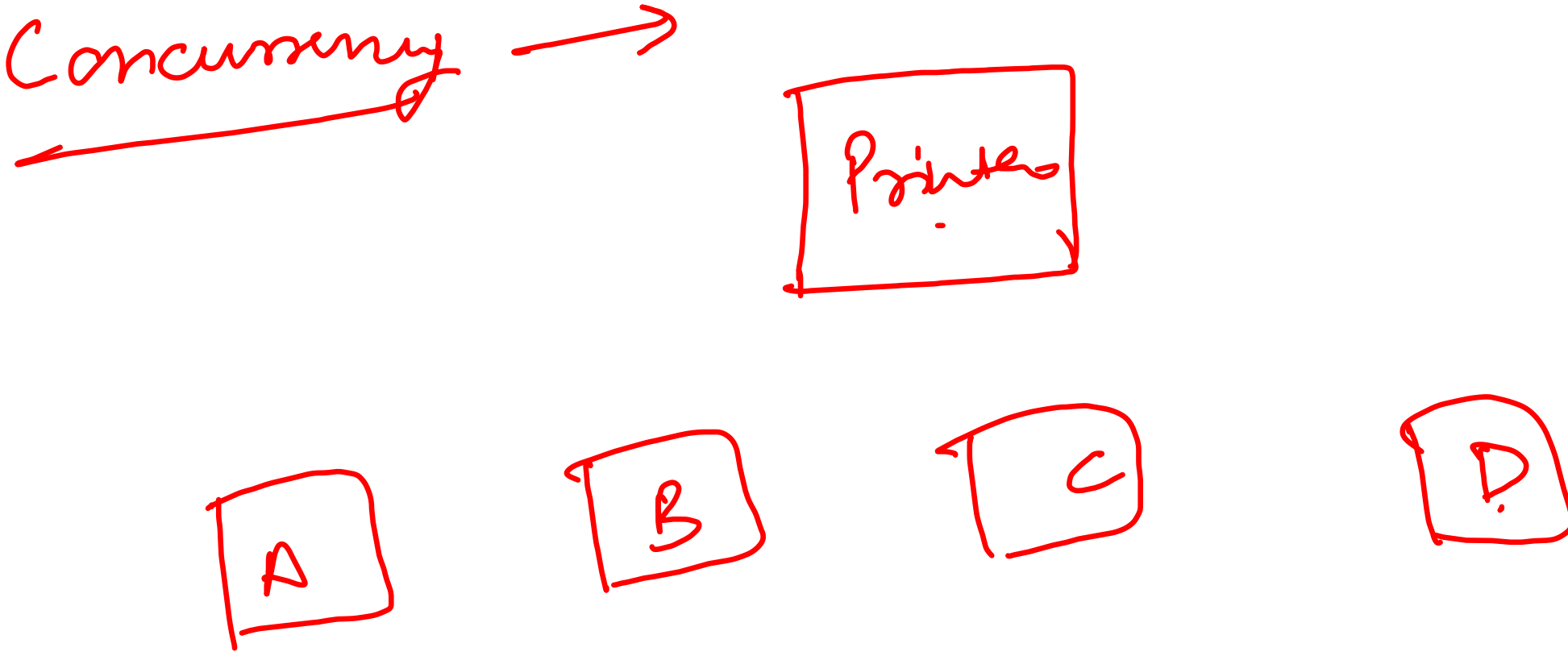
1. Non-atomic
2. Race conditions
3) Starvation
4) Deadlock

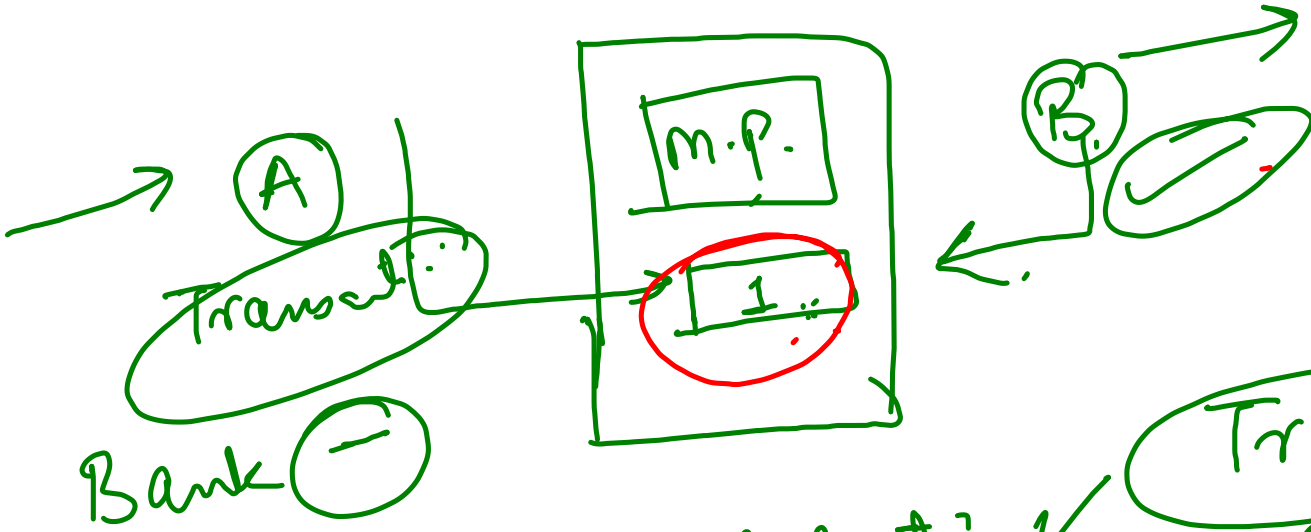
Deadlock






Atomicity
2 min
User-1
Transaction
1
0
Transacti-
1


Concurrency
Printer
A
B
C
D



User
Send

Seller
Rec X

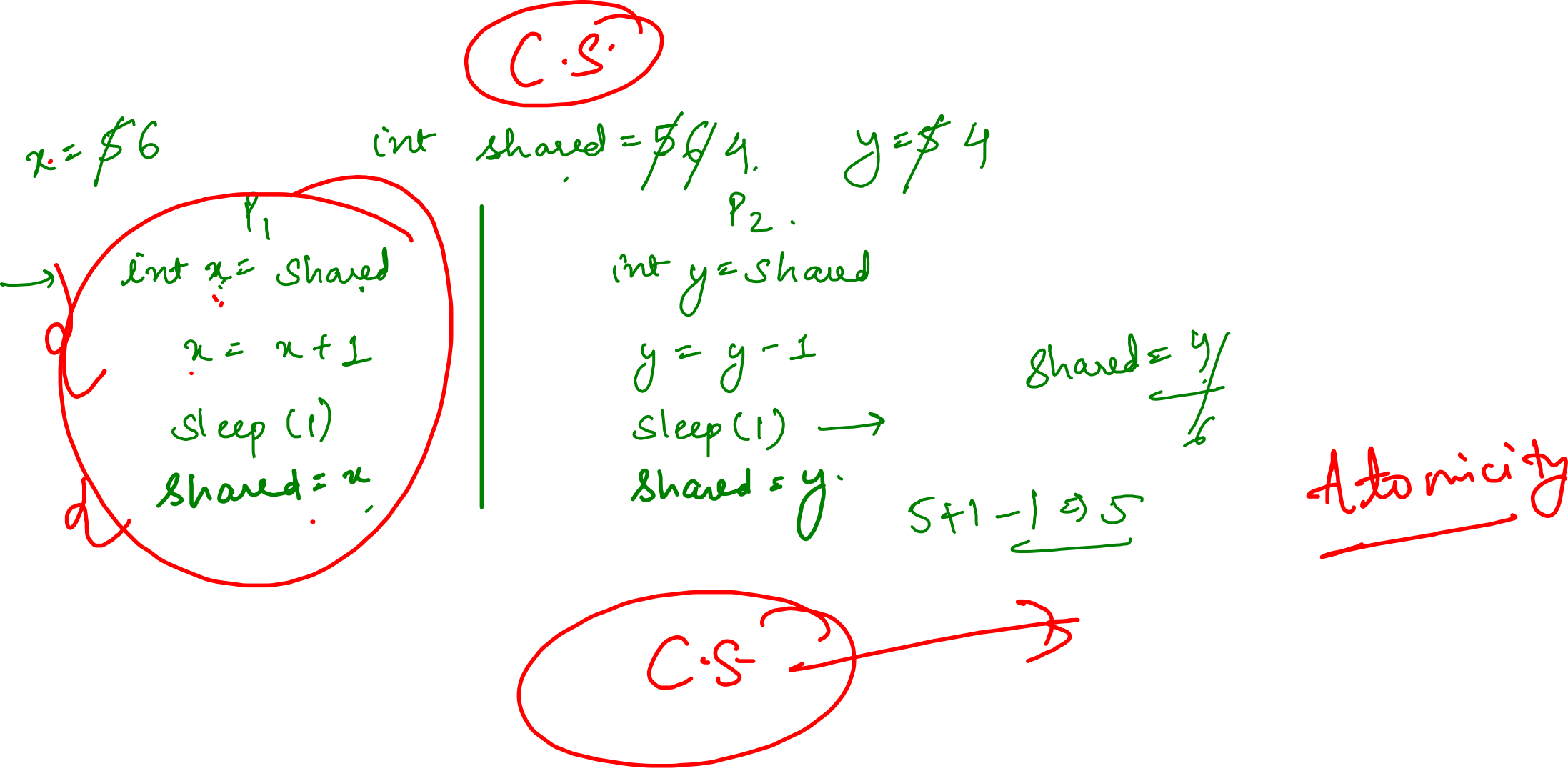
Transact
Confirm

1) Resource utilizati-✓

2) Shared resource
Race condition

Confirm

→ Send Recei


C.S.
x = 6
int shared = 6/4.
y = 4
P1
int x = shared
x = x + 1
sleep (1)
shared = x
P2.
int y = shared
y = y - 1
sleep (1)
shared = y.
shared = y
5+1-1 = 5
Atomicity
C.S.

count = 5.

PCB → registers

count ++

↙

count = count + 1

1) temp = count + 1

2) count = temp

temp = 6

count = count + 1

P1

temp = count + 1

count = temp

P2

count = count + 1

temp2 = count + 1

count = temp2

temp = 6

count = 6

count = $-$6

P_1

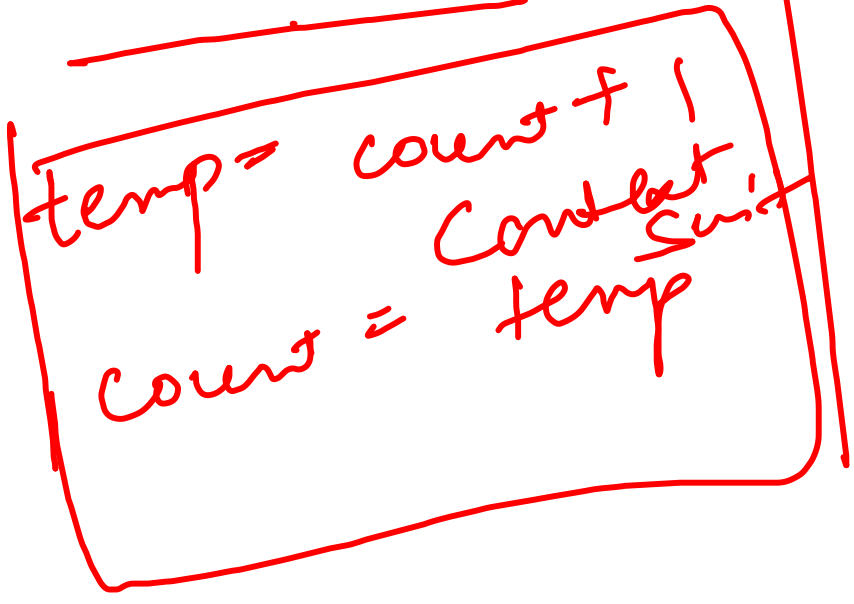
P_2 → High-pr

count++

temp = count + 1

count = temp

temp = 6



count++

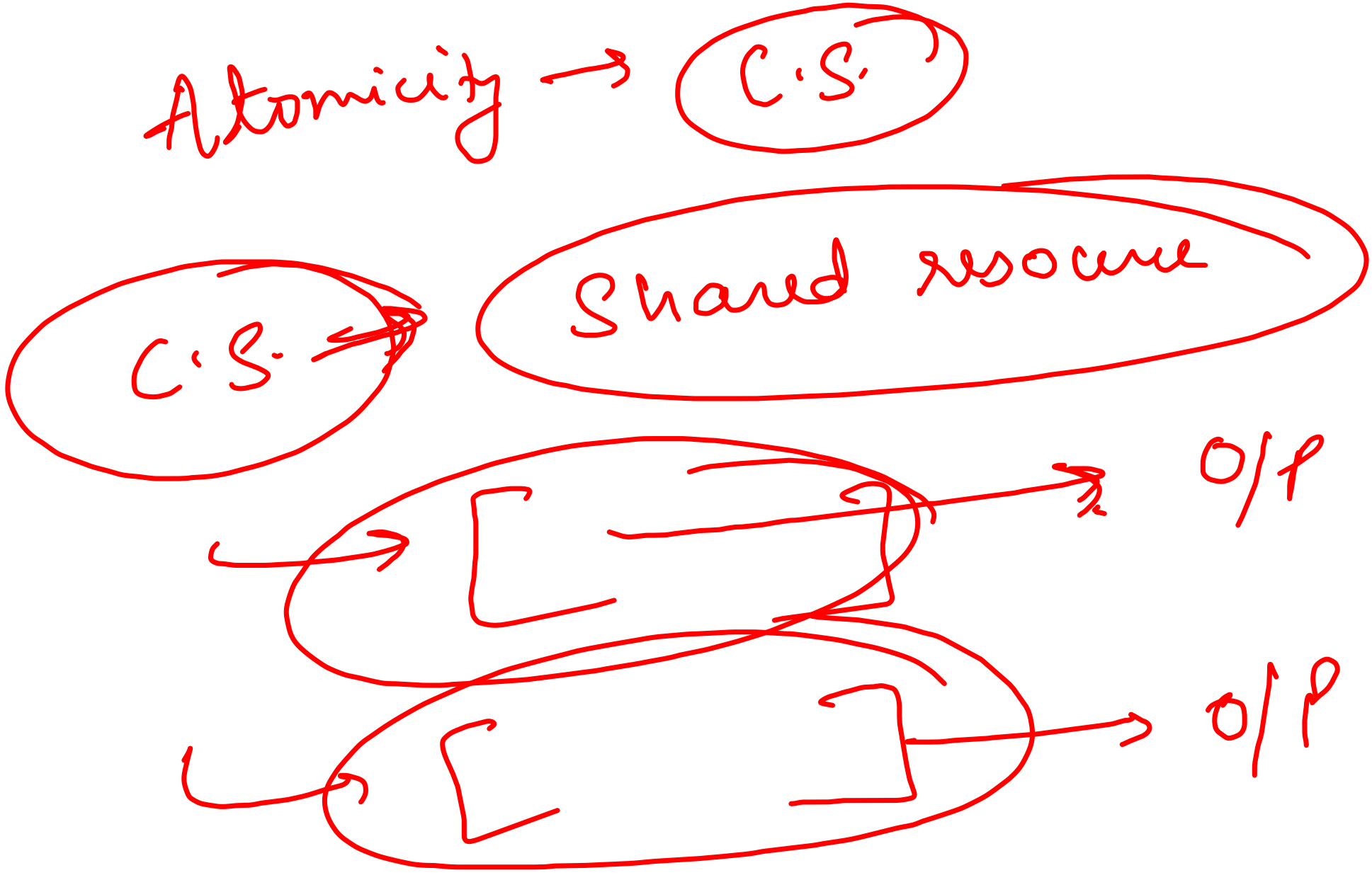
temp = count + 1

count = temp

temp = $-$6

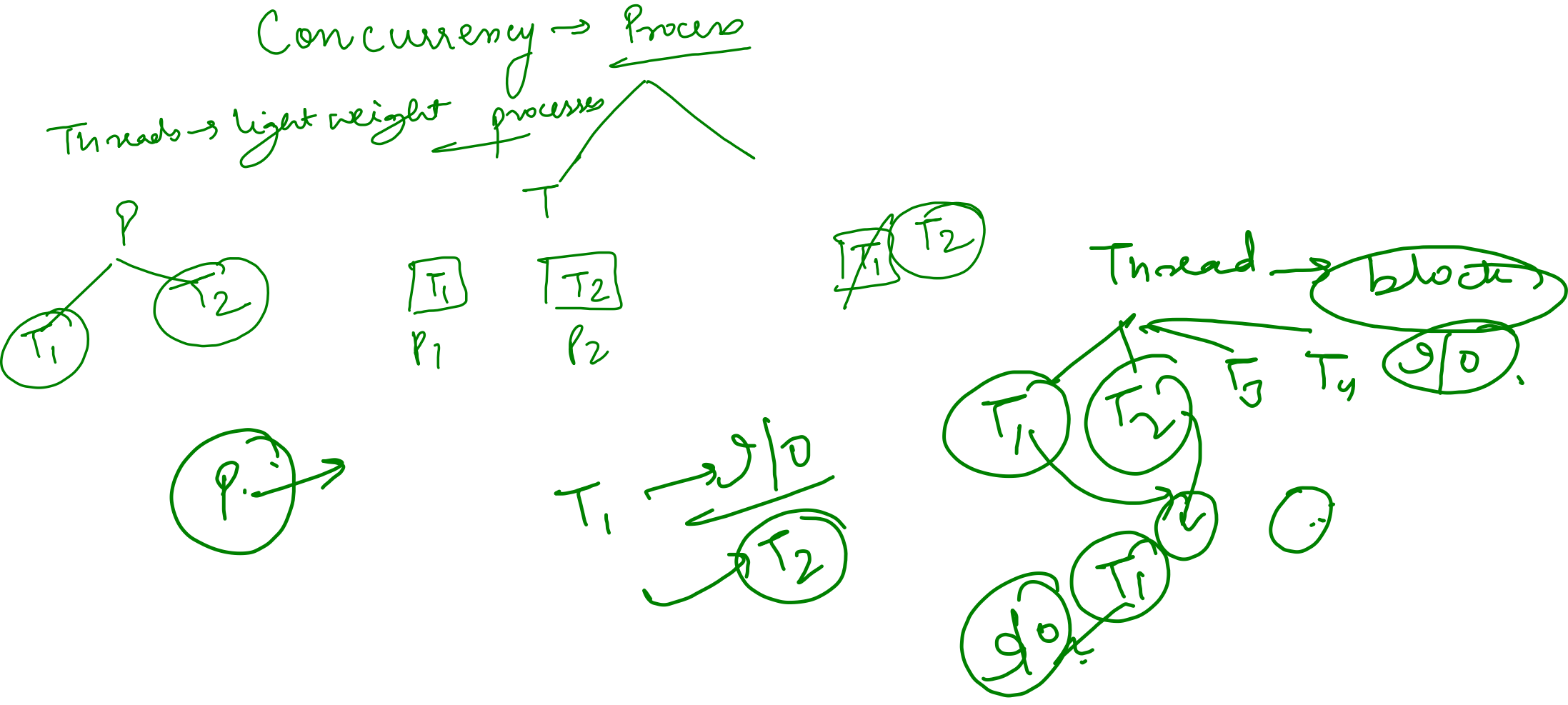
C

CO


Atomicity → C.S.
C.S.
Shared resource
O/P
O/P

1) Critical Section :- portion of the program where shared variables or shared resources are placed.

2) Race condition :- Condition when the final value depends upon the execution sequence of the process.


Concurrency → Process
Threads → light weight processes
T
P
T1
T2
P1
P2
T1
T2
Thread → block
T3
T4
I/O
T1
T2
T1
do

Conditions to achieve synchronization

Primary

1. **Mutual Exclusion** - No two processes can be simultaneously present inside the critical section.
2. **Progress** - No process running outside the critical section should block the other interested process from entering into the critical section, when C.S. is free.

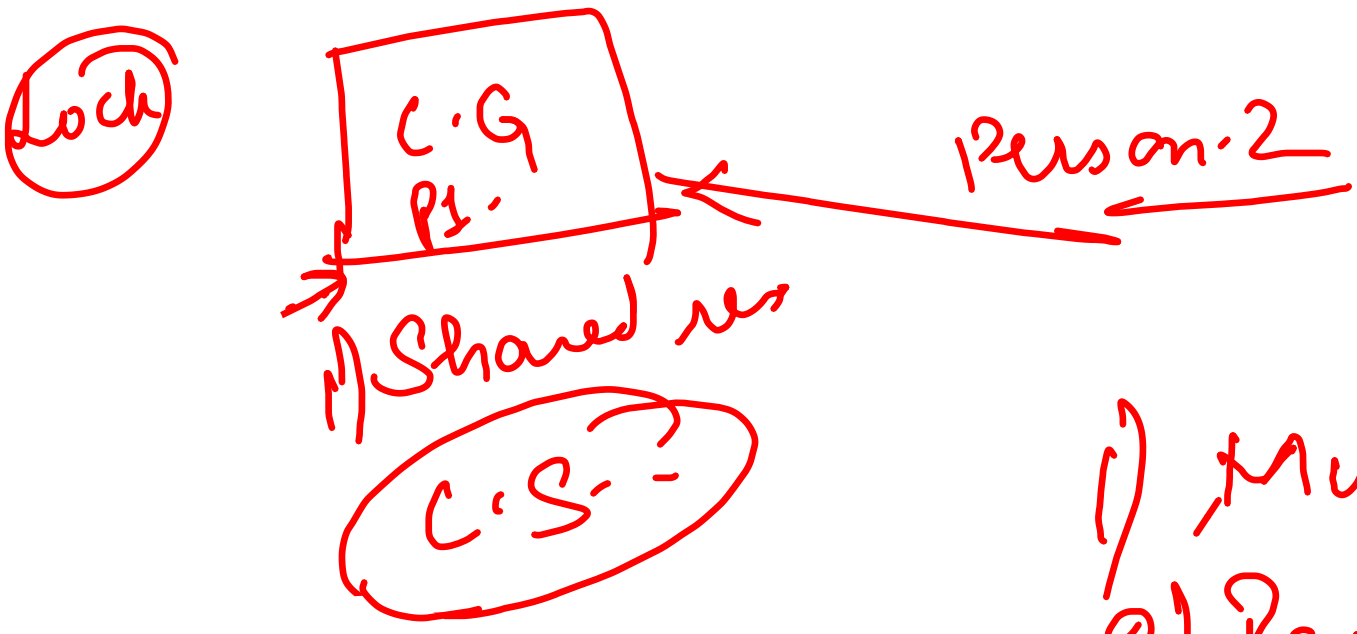
Sec.

per

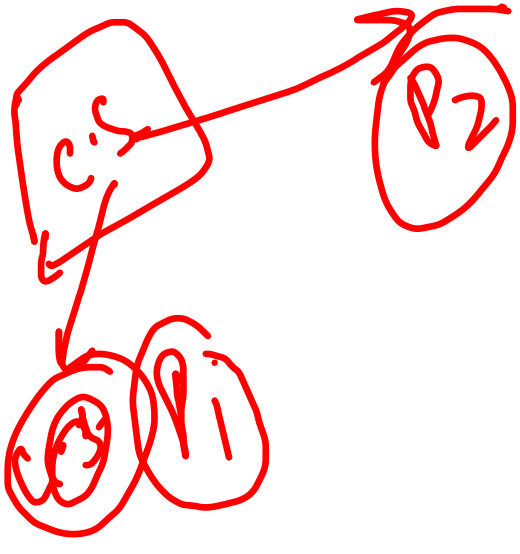
3. **Bounded waiting** - No process should have to wait forever to enter into the critical section.
4. **No assumption** - about H/W instructions, no. of processes, others.

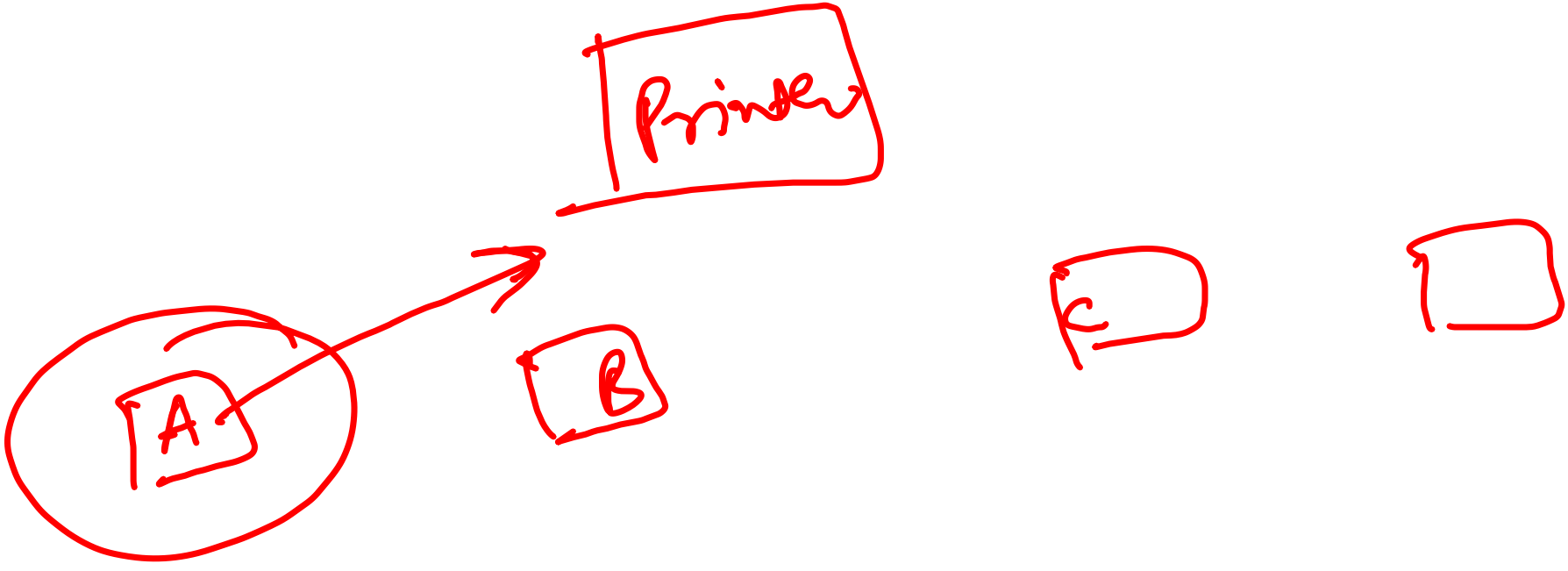
Synchronization

Process → IPC → Shar



1) Mutual exclusion
2) Progress
3) B.W




Printer
A
B
C

Synchronization Techniques

1) Lock variable
2) Semaphores

Process Po	Process P1
Entry section: while (turn! =0)	Entry Section: while (turn! =1)
Critical Section	Critical Section
Exit: Turn=1	Exit: Turn=0

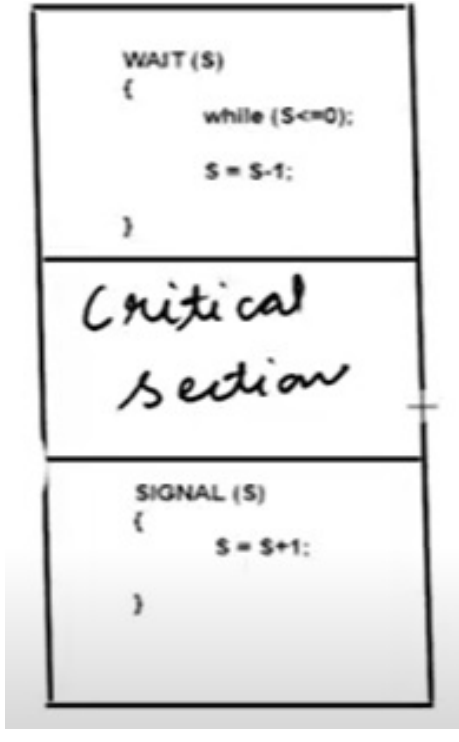
OS solutions :-

→ Semaphores →

- Binary
- Counting

Semaphores :- integer variables

Binary semaphore



Counting Semaphore

```
WAIT (S)
{
        S = S-1;

        if (S<0)
        {
          block;
        }


}
```

critical section

```
SIGNAL (S)
{
        S = S+1;

        if (S<=0)
        {
          wake up;
        }

}
```

1. Software solutions

1) Lock variable → n no. processes

2) Deckers Algo → 2 processes

3) Peterson's approach →

1. Load Ro, Lock
2. CMP #0, Ro
3. JNZ step 1
4. Store Lock, #1
   Critical Section
5. Store Lock, #0

1) TSL

1. Load Ro, Lock 2. CMP #0, Ro 3. JNZ Step 1 4. Store Lock, #1 Critical Section 5. Store Lock, #0	1. TSL Ro, m[Lock] 2. CMP #0, Ro 3. JNZ Step 1 Critical Section 4. Store Lock, #0

1) Peterson's algo-

2) Case studies
   1) Produce consum - pr
   2) Reader-writ
   3) Dini philosop