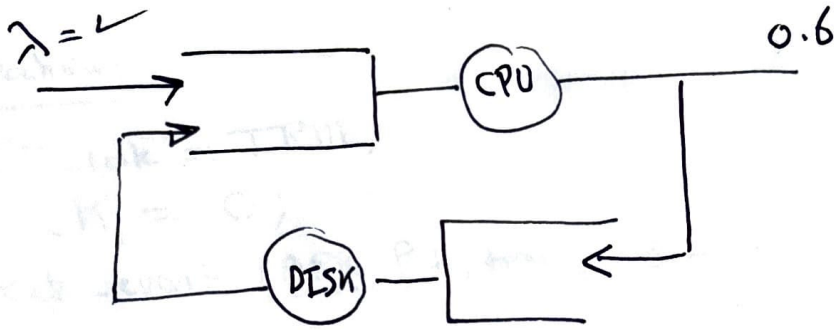
Project #1



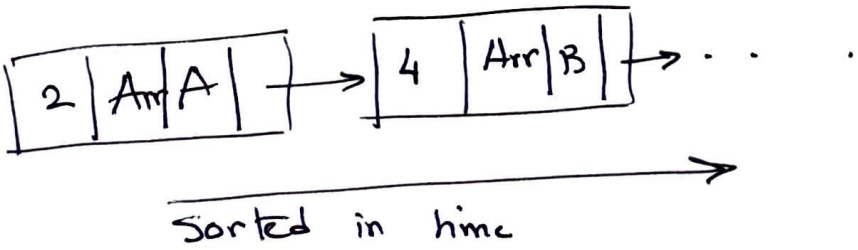
Discret-time event Simulation:

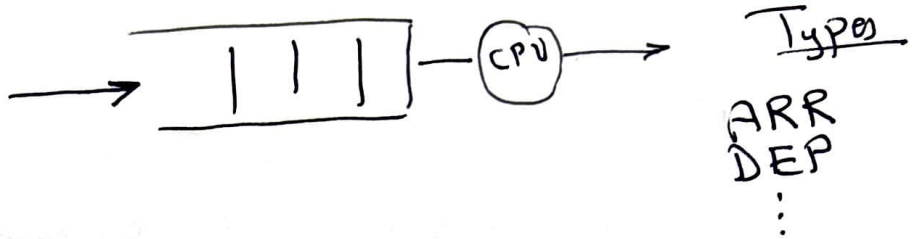
State

Events

Clock

Event Queue





Initialization:

```
CPU_idle = TRUE;
Clock = 0;
create_event (ARR, Pid, time)
```

← Function that creates an event and puts it in the Event Queue

Type	Pid	time
ARR	1	2.5

```
ReadyQueue = empty;
End_cond.
```

Running Engine

```
while( ! End_cond) {
    event = get_event ( )
    Clock = event.time;
    switch (event.type):
        case ARR:
            handle_ARR(event);

        case DEP:
            handle_DEP(event);
            ⋮
}
```

← Function that returns the first event in the Event Queue.

Output Results

λ 1 ⟶ 30

handle_ARR (↓ event arrived.) {

If (CPU_idle == TRUE)

Create_event (DEP, Pid, clock + service_time_rv);
CPU Idle = False;

else

Enqueue Pid into Ready Queue.

→ create_event (ARR, ~~Pid~~ new Process, clock + arrival time rv);

↓ Keeps the Simulation going

}

handle_DEP(↓ event DEP)

If Ready Queue == empty

CPU_idle = TRUE;

else

pop pid from Ready Queue.

Create_event (dep, Pid, clock + service_rv);

CPUidle = TRUE;

Clock = 0;

ReadyQ = NULL

E.Q

ARR	Pid ∅	2.5	

R.Q.

while (! End-cond) :

event

Clock = 2.5

Switch (ARR)

handle ARR → CPU Idle = FALSE;

E.Q

Serviching r.v = 3

DEP	Pid 0	5.5	

E.Q

ARR	Pid 1	4.5	