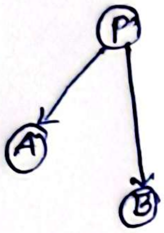
Summer 2025 OS Mid Solution

1(a)



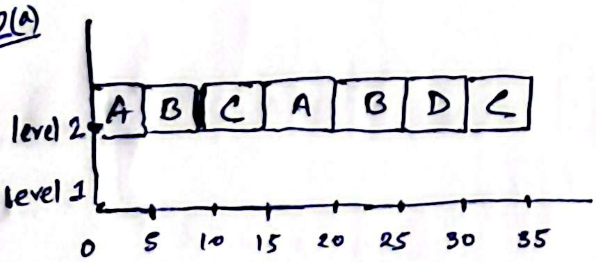
1(b) This is parent

3
2
1
0
This is a child
This is another child
This is another program
This is parent again.

1(c) * Shell figures out the path of the executable based on user command.

* Calls fork to create a child process to run the command.
* Calls some variant of exec to run the cmd.
* Waits for the command to complete by calling wait().

2(a)



Q2: ~~A, B, C, A, B, D, C~~

Q1: ~~A, B~~, ~~C~~

2(b) Avg. t. time = $\frac{20 + 25 + (30-14) + 35}{4}$

= 24 ms

2(c) Yes, because I/O bound processes have a small CPU burst and hence a higher fi.

3(a)

```
def uiu-shuttle-thread:
    lock(&m)
    shuttle_arrived = true
    int n = min(waiting_count, K)

    for(int i=1 ; i<=n; i++):
        signal(&cv_shuttle_arrived)
        wait(&cv_student_boarded, &m)

    waiting_count -= n
    shuttle_arrived = false
    depart()
    broadcast_signal(&cv_queue_open)
    unlock(&m)
```

3(b) Problem: Both threads can grab the lock.

Thread 1	Thread 2
while(*lock);	
Thread 1 Descheduled	
	while(*lock);
	*lock = true;
	Thread 2 Desched.
*lock = true	

Sol^n: TestAndSet function which works as an atomic instruction for checking availability of lock and acquiring it.

3(c) So that threads can have separate execution state and run independently.

2(d) No, processes with smaller CPU burst will get scheduled more number of times.

2(e) Yes, a CPU bound process with a low priority can starve if higher priority I/O bound processes keep arriving continuously.