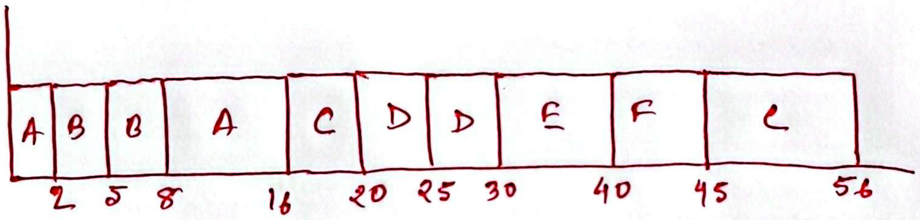
United International University
Course: Operating Systems (CSE 4509), Spring 2025
Class Test 2, Set D
Total Marks: 20, Time: 40 minutes

Name		ID		Section	

1. Draw the Gantt chart using STCF algorithm and find the average turnaround time and average response time from the following data: [6]

Process	A	B	C	D	E	F
Arrival (ms)	0	2	5	20	25	40
Duration (ms)	10	6	15	10	10	5



~~A - 8~~
~~B - 6 3~~
~~C - 15~~ 11
~~D - 10 5~~
~~E - 10~~

$$\text{Avg. turnaround time} = \frac{(16-0)+(8-2)+(56-5)+(30-20)+(40-25)+(45-40)}{6}$$
$$= 17.167$$

$$\text{Avg. response time} = \frac{0+0+(16-5)+0+(30-25)+0}{6} = 2.67$$

2. What are the differences between a Process and a Thread? [4]

Process	Thread
① A process is a running program	① A thread is a part of a process that can work as a process.
② Processes does not share its ~~virtual~~ virtual memory with other processes.	② Threads share the code segment and heap segment with other threads within a process.
③ A process communicates with other process through socket	③ A thread communicates wit other thread using shared space and ~~cont~~ synchronization primitive

3. Consider a scenario where a bus picks up waiting passengers from a bus stop periodically. The bus has a capacity of K. The bus arrives at the bus stop, allows up to K waiting passengers (fewer if less than K are waiting) to board, and then departs. Passengers have to wait for the bus to arrive and then board it. Passengers who arrive at the bus stop after the bus has arrived should not be allowed to board, and should wait for the next time the bus arrives. The bus and passengers are represented by threads in a program. The passenger thread should call the function board() after the passenger has boarded and the bus should invoke depart() when it has boarded the desired number of passengers and is ready to depart. The threads share the following variables, none of which are implicitly updated by functions like board() or depart(). Below is given synchronized code for the passenger thread. You should not modify this in any way.

Synchronization Code for passenger thread

```
mutex // lock variable
bus_arrived_cond, passenger_boarded_cond // conditional variables
int waiting_count = 0
bool bus_arrived = false

function passenger_thread():
  lock(&mutex)
  waiting_count++
  while not bus_arrived:
    wait(&bus_arrived_cond, &mutex)
  board() // Board the bus
  signal(&passenger_boarded_cond)
  unlock(&mutex)
```

Write down the corresponding synchronized code for the bus thread that achieves the correct behavior specified above. The bus should board the correct number of passengers, based on its capacity and the number of those waiting. The bus should correctly board these passengers by calling wait/signal functions suitably. The bus code should also update the waiting count as required. Once boarding completes, the bus thread should call depart(). You can use any extra local variables in the code of the bus thread, like integers, loop indices and so on. However, you must not use any other extra synchronization primitives (lock or conditional variables). **[10]**

```
function bus_thread():
    lock(&mutex)
    bus_arrived = true
    int N = min(K, waiting_count)
    for i in 1 to N:
        signal(&bus_arrived_cond)
        wait(&passenger_boarded_cond, &mutex)
    waiting_count -= N
    bus_arrived := false
    unlock(&mutex)
    depart()
```