COMSATS University Islamabad, Lahore Campus

Midterm Examination – Spring 2024 (Solution)

Course Title:	Principles of Operating Systems			Course Code:	CSC 323	Credit Hours:	03
Course Instructor/s:	Mr. Nadeem Ghafoor Ch., Ms. Zeenat Afzal. Mr. Mudassar			Program Name:	BSCS, BSSE, BSEE, BSCE		
Semester:	5th	Batch:		Section:		Date:	
Time Allowed:	**90 Minutes**			**Maximum Marks:**		**25**	
Student's Name:				Reg. No.			

Important Instructions / Guidelines:

- Write to the point and avoid unnecessary details

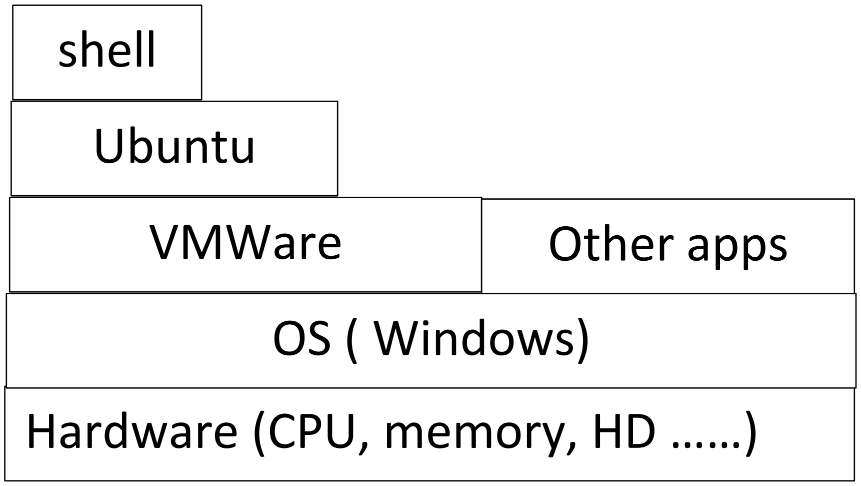
Q. No 1: **CLO: <1>: Bloom Taxonomy Level: <Understanding>** **(2+3+3=8 marks)**

a. Modern CPUs have two execution modes i.e. kernel mode and user mode, why we need the two modes?

The kernel mode can use all general instructions as well as privileged instructions. Privileged instructions help the CPU access sensitive information (e.g., clear the cache) and carry out vital operations (e.g., I/O). On the other hand, only general instructions are available in the user mode. If a privileged instruction is executed in the user mode, the CPU treats the instruction as an illegal one and traps to the operating system. An operating system usually runs in the kernel mode so that it can have access to all hardware components and no user can affect the execution of the OS. This is for protection purpose. If a normal program has to execute privileged instructions, it does so by system call.

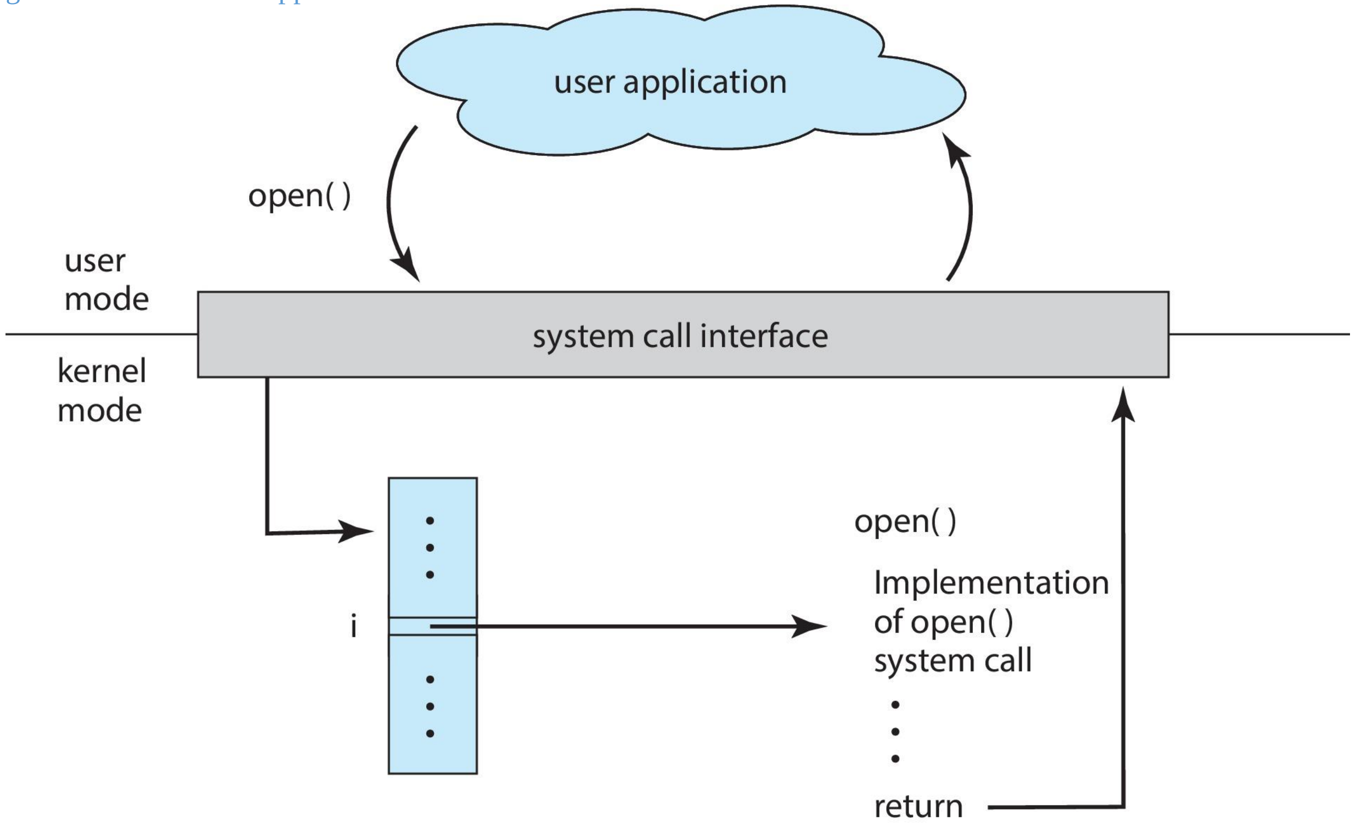
b. What is a virtual machine and what are its benefits, give an example.

A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware. The "host" operating system creates an illusion for each "guest" operating system that it is running on real hardware. Fundamentally, multiple execution environments (different operating systems) can share the same hardware. It is useful for development and testing. For example, we use VMWare in our labs to run Ubuntu on Windows. VMWare provides the virtual environment on which we run Ubuntu without disturbing our base operating system and other applications.



c. Using an example explain how a normal user program executes a system call?

A normal user program will use an API to access a system call. For example, if we have to open a file we can use fopen in C. This C library function calls open() system call. In order to execute open, a trap instruction gets executed which performs two actions simultaneously. It switches the mode bit from user mode to kernel mode and jumps to the location of open() function. This jump location/address is stored in system call table. As a result the execution of the program leaves the user application and executes instructions of the open() system call which are part of the OS. Upon completion of open() the execution goes back to the user application and the mode bit is switched back to user mode.



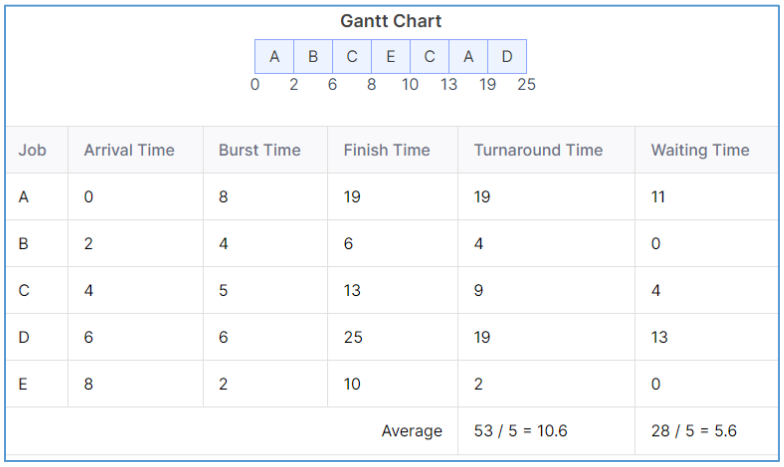
Q. No 2: CLO: <2>: Bloom Taxonomy Level: <Analyzing> (2+6=8 marks)

a. What is the difference between waiting time and response time? Under what situation(s) are these equal?
Response time is the amount of time it takes for the CPU to respond to a request made by a process. It is the duration between the arrival of a process and the first time the process gets the CPU. Waiting time is the amount of time that a process is in the ready queue waiting for the CPU, i.e., from its arrival until it exits the system. Response time is equal to waiting time in all non-preemptive scheduling algorithms

b. Given the following arrival time, burst time and Priority for each process, draw the **Gantt chart** and compute the **average waiting time** for the following CPU scheduling algorithms.

Note: (Do not consider context switch time and resolve the clash (if any) on the basis of FCFS).

- Shortest Remaining Time First



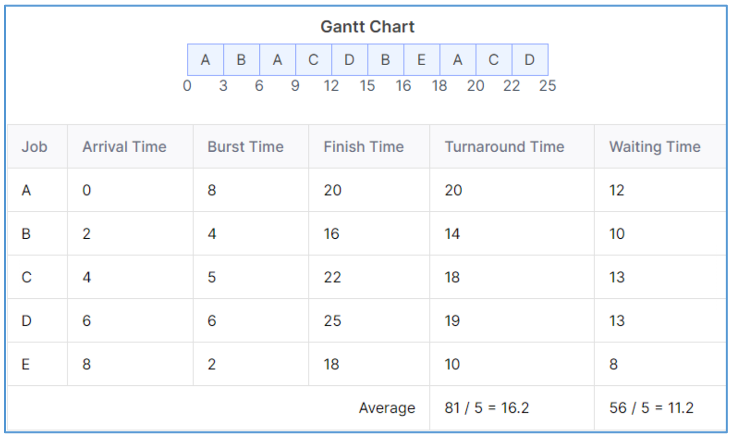
Gantt Chart

A	B	C	E	C	A	D

0 2 6 8 10 13 19 25

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
A	0	8	19	19	11
B	2	4	6	4	0
C	4	5	13	9	4
D	6	6	25	19	13
E	8	2	10	2	0
			Average	53 / 5 = 10.6	28 / 5 = 5.6

- Round-Robin with time quantum of 3



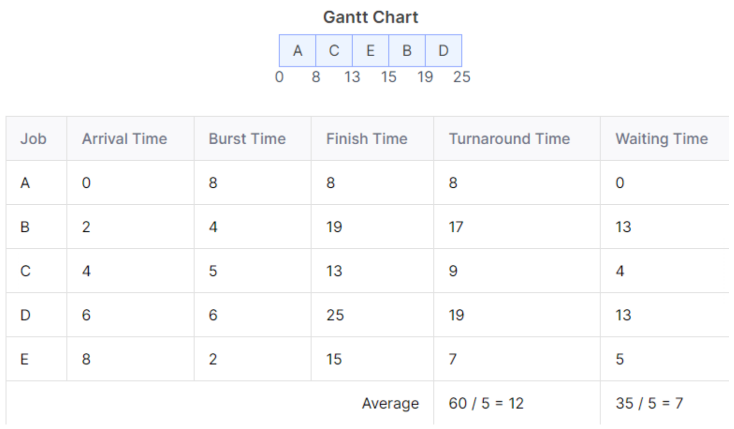
Gantt Chart

A	B	A	C	D	B	E	A	C	D

0 3 6 9 12 15 16 18 20 22 25

Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
A	0	8	20	20	12
B	2	4	16	14	10
C	4	5	22	18	13
D	6	6	25	19	13
E	8	2	18	10	8
			Average	81 / 5 = 16.2	56 / 5 = 11.2

- Non - Preemptive Priority (A lower number indicates a greater priority)



Gantt Chart

A	C	E	B	D

0 8 13 15 19 25

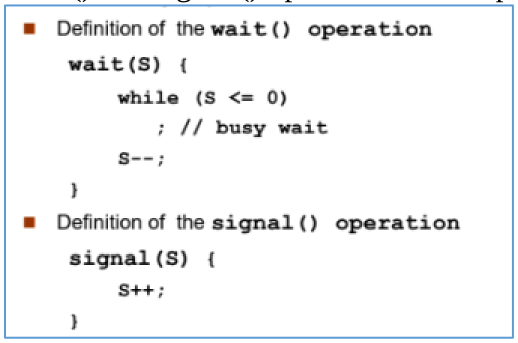
Job	Arrival Time	Burst Time	Finish Time	Turnaround Time	Waiting Time
A	0	8	8	8	0
B	2	4	19	17	13
C	4	5	13	9	4
D	6	6	25	19	13
E	8	2	15	7	5
			Average	60 / 5 = 12	35 / 5 = 7

Q.No 3: CLO: <2>: Bloom Taxonomy Level: <Analyzing> (2+2+5= 9marks)

a. Differentiate between binary and counting semaphore?

Binary semaphores take on only two values which are 0 and 1, and are used to achieve mutex (mutual exclusion). On the other hand, counting semaphores can hold any nonnegative integer value and are used to control access to a resource with multiple instances.

b. Explain the implementation of wait () and signal () operation in semaphore?



- Definition of the `wait() operation`

```
wait(S) {
    while (S <= 0)
       ; // busy wait
    S--;
}
```

- Definition of the `signal() operation`

```
signal(S) {
    S++;
}
```

c. Select the usage of wait() and signal() to synchronize processes A, B, C, D, E and F by using semaphores so that process A must finish executing before B starts, process B must finish before C or D starts, and process D must finish before process E or F starts. F should finish last. Show your solution. You should assume three semaphores **S1**, **S2, S3 and S4** and all **initialized to zero**. Note: All processes must execute once.

Process A	Process B	Process C	Process D	Process E	Process F

Print (A)
Signal(S1)
	Wait(S1)
	Print(B)
	Signal(S2)
		Wait(S2)
		Print(C)
		Signal(S2)
		Signal(S4)
			Wait(S2)
			Print(D)
			Signal(S2)
			Signal(S3)
				Wait(S3)
				Print(E)
				Signal(S4)
					Wait(S4)
					Wait(S4)
					Print(D)