University of Asia Pacific

Department of CSE

Mid-Semester Examination, Fall 2020

Name: Rashik Rahman

Reg ID: 17201012

Year: 4th

Semester: 1st

Course Code: CSE 405

Course Title: Operating Systems

Date: 24.02.2021

"During Examination and upload time I will not take any help from anyone. I will give my exam all by myself."

University of Asia Pacific

Admit Card

Mid-Term Examination of Fall, 2020

Financial Clearance	PAID

Registration No : 17201012
Student Name : Rashik Rahman
Program : Bachelor of Science in Computer Science and Engineering

SI.NO.	COURSE CODE	COURSE TITLE	CR.HR.	EXAM. SCHEDULE
1	CSE 400	Project / Thesis	3.00	
2	CSE 330	Industrial Training	1.50	
3	CSE 401	Mathematics for computer Science	3.00	
4	CSE 403	Artificial Intelligence and Expert Systems	3.00	
5	CSE 404	Artificial Intelligence and Expert Systems Lab	1.50	
6	CSE 405	Operating Systems	3.00	
7	CSE 406	Operating Systems Lab	1.50	
8	CSE 407	ICTLaw, Policy and Ethics	2.00	
9	CSE 410	Software Development	1.50	
10	CSE 427	Topics of Current Interest	3.00	

Total Credit: 23.00

1. Examinees are not allowed to enter the examination hall after 30 minutes of commencement of examination for mid semester examinations and 60 minutes for semester final examinations.

2. No examinees shall be allowed to submit their answer scripts before 50% of the allocated time of examination has elapsed.

3. No examinees would be allowed to go to washroom within the first 60 minutes of final examinations.

4. No student will be allowed to carry any books, bags, extra paper or cellular phone or objectionable items/incriminating paper in the examination hall.
Violators will be subjects to disciplinary action.

This is a system generated Admit Card. No signature is required.

Admit Card Generation Time: 21-Feb-2021 10:23 PM

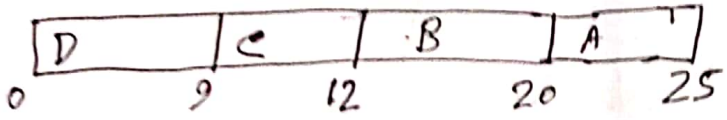
17201012

(1)

Answer to the Q.NO.16(a)

First come first served:

Gearttchart:



Average wait time:

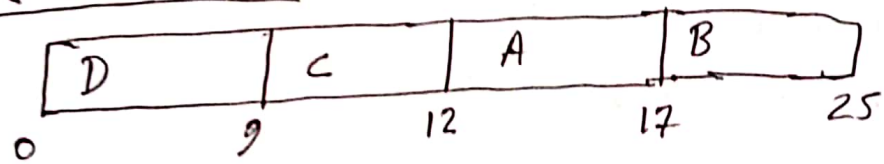
Average wait time = $\frac{0+9+12+20}{4} = 10.25$

$= \frac{0+(9-1)+(12-5)+(20-6)}{4} = \frac{0+8+7+17}{4}$

$= 7.25$

Shortest Job First

Gantt chart:



wait time for D = 0

wait time for C = 9 - 1 = 8

wait time for A = 12 - 6 = 6

wait time for B = 17 - 5 = 12

∴ average wait time = $\frac{0+8+6+12}{4} = 6.5$

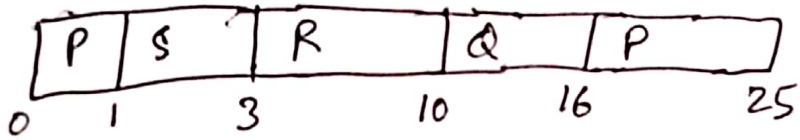
17201012

(2)

Answer to the Q. No. 2(a)

Priority Scheduling:

Gantt chart:



wait time for P = 0 - 0 + (16-1) = 15

wait time for S = 1 - 1 = 0

wait time for R = 3 - 3 = 0

wait time for Q = 10 - 2 = 8

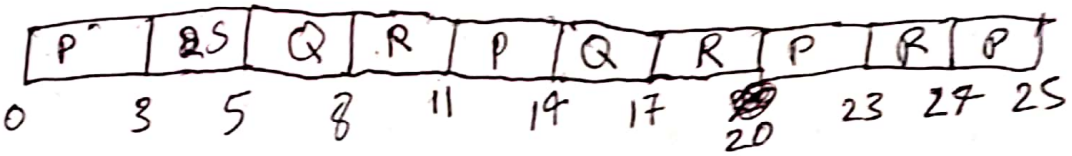
$\therefore$ average wait time $= \frac{15+0+0+8}{4} = 5.75$

17201012
(3)

Round Robin:

Here,
quantum = 3

Gantt chart :



wait time = Last wait time - (count of process - 1) x quantum
~~average wait time~~ = or, last wait time (if burst time ≤ quantum) - arrival time

wait time for P = 24 - 3x3 - 0 = 15

wait time for S = 3 - 1 = 2

wait time for Q = 14 - 1x3 - 2 = 9

wait time for R = 23 - 2x3 - 3 = 14

∴ average wait time = $\frac{15+2+9+14}{4} = 10$

17201012

(4)

Answer to the Q.NO. 4(a)

Whenever a computer starts the first process to run is init and all the other processes are it's child, furthermore child process can also have their child. A child process is created by the means of fork() function. For each process regardless of child or parent a memory image is created and stored in memory. After execution of each process this memory image still remains in memory, thus these process are called zombie process and for these zombie processes memory will be full causing the OS to crash or suffer from hang issues. But there exist a function called wait() which can prevent the zombie effect on memory fillup effect. When we call wait() function parent process waits untill child process is executed the parent is executed and the memory image is cleared from the memory. Thus we can prevent the zombie effect on memory full case.

17/2/2012

(5)

Answer to the Q. No. 4(b)

A pipe has two file descriptons. One is read another one is write. A regular pipe is where these both file descriptons are in a single process. In this case when fork() is called child inherits all properties of parent including the pipe. So parent and child shares these two file descriptons after fork(). For this reason parent and child can have more efficient way of inter process communication. Here Parent can use write file descripton and child can use read file descripton also vice-versa to communicate.