Operating System

B.Sc (H) III Sem | Computer Science - S.No 1133 | UPC : 2342012302

Solutions

Section A

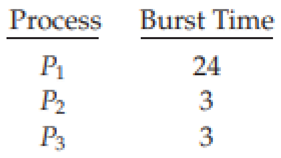
Q1 a (2 marks – 1 each for zombie and orphan process)

Orphan Process	Zombie Process
If a parent does not invoke wait() and instead terminates, thus leaving its child processes as orphans.	A process that has terminated, but whose parent has not yet called wait(), is known as a zombie process. All processes transition to this state when they terminate, but generally they exist as zombies only briefly.

Q1 b (1+ 1 for example)

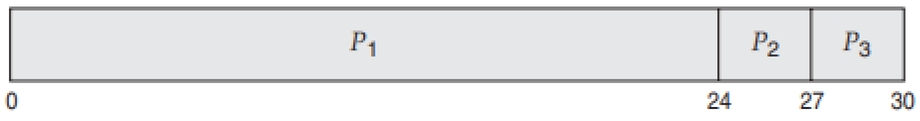
The Convoy Effect is a phenomenon associated with the First Come First Serve (FCFS) algorithm, in which the whole Operating System slows down due to a few slow processes.
Suppose there is one CPU-intensive (large burst time) process in the ready queue, and several other processes with relatively fewer burst times but are Input/Output (I/O) bound (Need I/O operations frequently). All the I/O processes end up waiting in the ready queue until the CPU-bound process is done. There is a convoy effect as all the other processes wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if the shorter processes were allowed to go first.

For example for processes (Any example and its explanation with or without Gantt Chart would be fine)



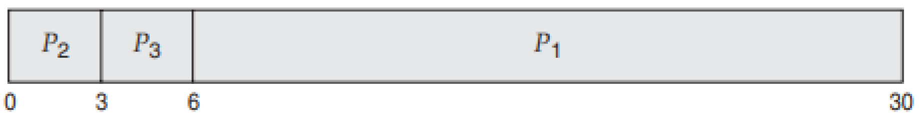
Process	Burst Time
P_1	24
P_2	3
P_3	3

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart,



The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3. Thus, the average waiting time is (0 + 24 + 27)/3 = 17 milliseconds.

But If the processes arrive in the order P2, P3, P1, however, the results will be as shown in the following Gantt chart:



The average waiting time is now (6 + 0 + 3)/3 = 3 milliseconds. This reduction is substantial.

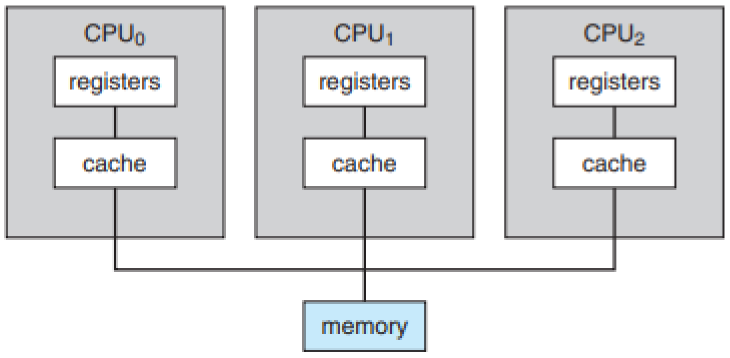
Q1 c (2 marks)

Hello
Hello
World (in any order)

Or Hello
World

Q1 d (1+ 1 for example)

In a multiprocessor environment where, in addition to maintaining internal registers, each of the CPUs also contains a local cache . In such an environment, a copy of A may exist simultaneously in several caches. Since the various CPUs can all execute in parallel, we must make sure that an update to the value of A in one cache is immediately reflected in all other caches where A resides. This situation is called cache coherency, and it is usually a hardware issue (handled below the operating-system level). For example.



Q1 e (2 marks)

Paging is implemented by breaking up an address into a page and offset number. It is most efficient to break the address into X page bits and Y offset bits, rather than perform arithmetic on the address to calculate the page number and offset. Because each bit position represents a power of 2, splitting an address between bits results in a page size that is a power of 2.

Q1f This question is out of syllabus. Please give full 2 marks to those who have attempted.

(1+1 for each)

Asymmetric multiprocessing	Symmetric multiprocessing
Each processor is assigned a specific task. A	Each processor performs all tasks within the

boss processor controls the system; the other processors either look to the boss for instruction or have predefined tasks. This scheme defines a boss–worker relationship. The boss processor schedules and allocates work to the worker processors.	operating system. SMP means that all processors are peers; no boss–worker relationship exists between processors.

Q1g (1/2 marks each)

i. No
ii. Yes
iii. Yes
iv. No

Q1h (1+1 marks each)

Privileged instructions- any example of I/O control, timer management, and interrupt management etc.
Instructions that can run in user mode - Reading the system time, Reading the status of the processor, Sending the final printout of a printer, etc.

Q1i (1+ 1 for example)

Data parallelism means distributing subsets of the same data across multiple computing cores and performing the same operation on each core. Consider, for example, summing the contents of an array of size N. On a single-core system, one thread would simply sum the elements [0] . . . [N − 1]. On a dual-core system, however, thread A, running on core 0, could sum the elements [0] . . . [N/2 − 1] while thread B, running on core 1, could sum the elements [N/2] . . . [N − 1]. The two threads would be running in parallel on separate computing cores. (Any example would do.)

Q1j (2 marks) (deduct 1 marks if explanation is not given)

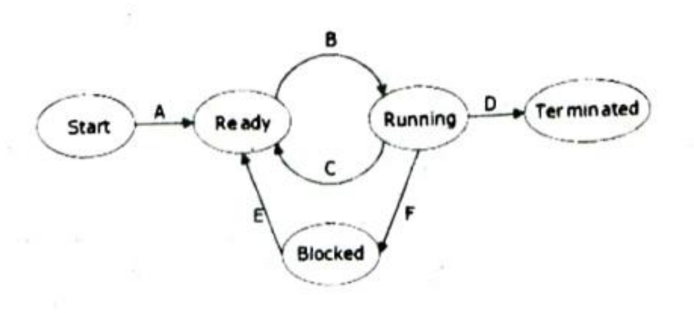
Relocation register = 600
Limit register = 250
Logical address = 200
Since 200 < 250 so corresponding physical address = 600 + 200 = 800

Q1 k (1+1 for each difference)

Monolithic	Microkernel (any one)
The monolithic approach is a traditional software design model where a single code base is used to build an application as a unified unit. This approach can be useful for simple applications or prototypes, but it can become challenging as the application grows more complex.	This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel. There is little consensus regarding which services should remain in the kernel and which should be implemented in user space
	Typically, however, microkernels provide

	minimal process and memory management, in addition to a communication facility
	The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through message passing

Q1 l (2+2) (if justification not written deduct 1 marks each)



i. No – termination of process (Transition D) will result in short term scheduler to select process from ready queue and move to running state. Since long term scheduling controls degree of multiprogramming it will have no effect on transition A.
ii. Yes – Blocked state process can move to ready state without affecting process in running state.

Q1 m (1+1+1+1 each)

i. Long term scheduler
ii. Kernel
iii. Seek time
iv. Context switch

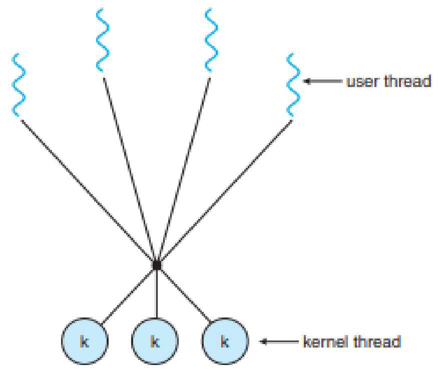
Section B

Q2 a (4 marks-please don't cut marks for calculation) (deduct 1 marks for not writing formula)

effective access time = (1 − p) × ma + p × page fault time
350 = (1-p) × 200 + p × 5000000
350= 200-200p+5000000p
150=4999800p
p<0.0000300012

Maximum page fault 106 memory accesses for every 1 page fault

Q2 b (3marks – 1 marks each for diagram, advantage, disadvantage)



Advantage (Any one)
Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor. Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.

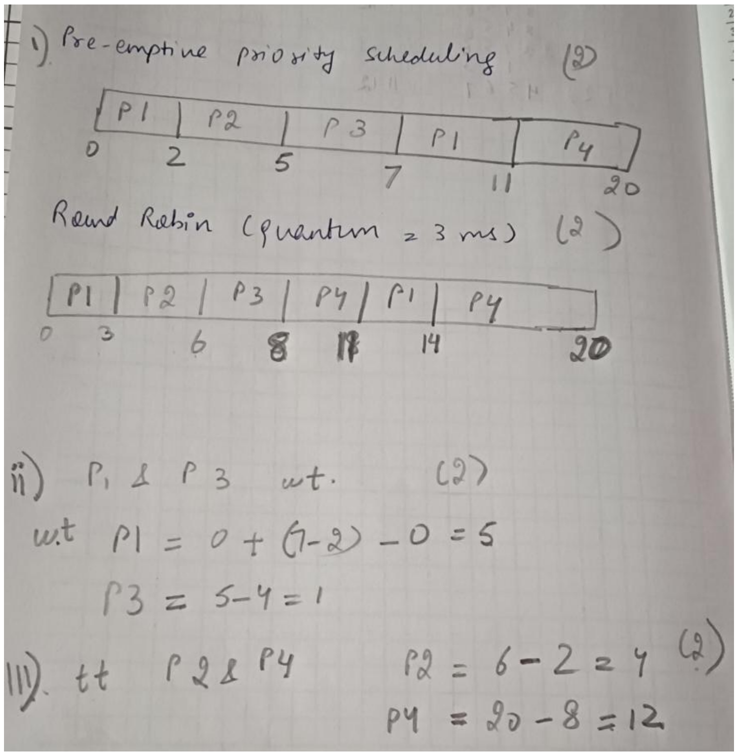
Disadvantage (Any one)

i. Complex Thread Management:
In a many-to-many model, the operating system kernel is required to manage multiple user-level threads and assign them to kernel threads. This can lead to significant complexity in managing these threads, especially when the system scales up with numerous threads.

ii. Overhead from Synchronization:
Synchronizing between user-level threads and kernel threads introduces additional overhead

Q2 c (4+2+2)

Q3 a


i) Pre-emptive priority scheduling (2)

P1	P2	P3	P1	P4

0 2 5 7 11 20

Round Robin (quantum = 3 ms) (2)

P1	P2	P3	P4	P1	P4

0 3 6 8 11 14 20

ii) P_1 & P_3 wt. (2)

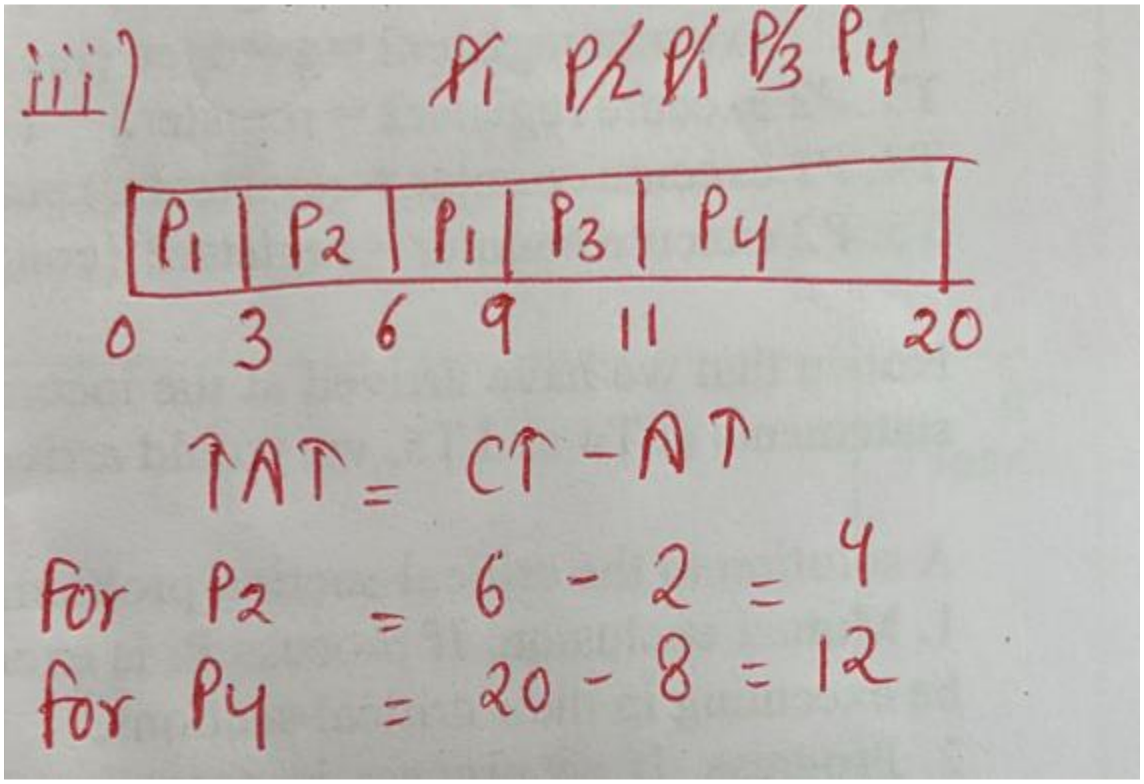
w.t $P1 = 0 + (7-2) - 0 = 5$

$P3 = 5-4 = 1$

iii). tt $P2$ & $P4$ $P2 = 6-2 = 4$ (2)

$P4 = 20-8 = 12$

Or


iii) ~~P_1~~ ~~P_2~~ ~~P_1~~ ~~P_3~~ P_4

P_1	P_2	P_1	P_3	P_4

0 3 6 9 11 20

TAT = CT − AT

for P_2 = 6 − 2 = 4

for P_4 = 20 − 8 = 12

(1 marks each)

i. Line A – 0
ii. Line B - 1500
iii. Line C - 1567
iv. Line D – 1500

Q3 b (2+3)

P1: counter++
P2: counter –

P1: counter++
Note that the statement "counter++" may be implemented in machine language as follows:

```
register1 = counter
register1 = register1 + 1
counter = register1
```

P2: counter –
Similarly, the statement "counter--" is implemented as follows:

```
register2 = counter
register2 = register2 − 1
counter = register2
```

The concurrent execution of P1:"counter++" and P2:"counter--" is equivalent to a sequential execution which may be interleaved in some arbitrary order. One such interleaving is the following:

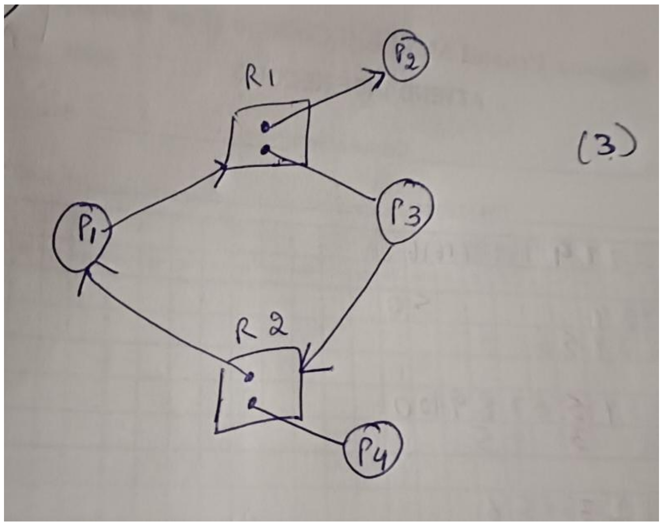
T0: P1 execute register1 = counter {register1 = 10}
T1: P1 execute register1 = register1 + 1 {register1 = 11}
T2: P2 execute register2 = counter {register2 = 10}
T3: P2 execute register2 = register2 − 1 {register2 = 9}
T4: P1 execute counter = register1 {counter = 11}
T5: P2 execute counter = register2 {counter = 9}

Notice that we have arrived at the incorrect state "counter == 9", If we reversed the order of the statements at T4 and T5, we would arrive at the incorrect state "counter == 11".

A solution to the critical-section problem must satisfy the following three requirements:
1. Mutual exclusion. If process Pi is executing in its critical section, then no other processes can be executing in their critical sections.
2. Progress. If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

3. Bounded waiting. There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

Q3 c (3+3)



There is no deadlock. (1 mark)
That process P4 may release its instance of resource type R2. That resource can then be allocated to P3, breaking the cycle. In summary, if a resource-allocation graph does not have a cycle, then the system is not in a deadlocked state. If there is a cycle, then the system may or may not be in a deadlocked state. (2 for justification)

Q4	a)	Logical address space=256 pages Page size=1024 bytes = 2^{10} Physical address space=128 frames i) No. of bits in logical address= 256 X 1024 = 2^8 X 2^{10} = 2^{18} = 2^m m=18 bits ii) No. of bits in physical address=128 X 1024 = 2^7 X 2^{10} = 2^{17} = 2^n n=17 bits	2 + 2 marks
	b)	Operations of a directory (ANY TWO) • Search for a file • Create a file • Delete a file • List a directory • Rename a file • Traverse the file system Disadvantage of a two-level directory structure: • Cannot create sub-directory	2 + 3 marks

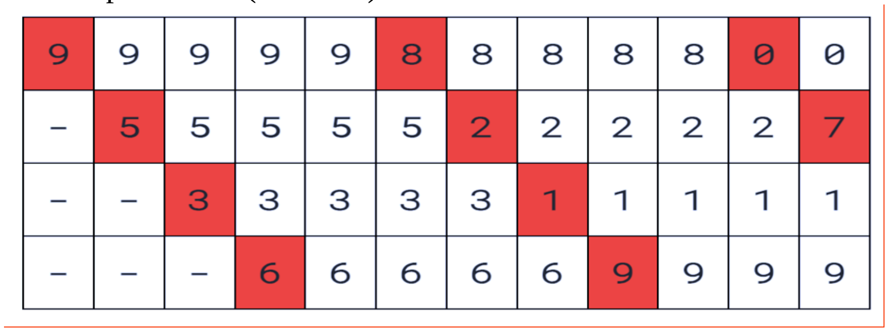
Resolution by tree-structured directory

- Directory structure allows sub directories to be created

c) Page reference string:
9, 5, 3, 6, 5, 8, 2, 1, 9, 9, 0, 7

3+3 marks (Give 1 marks if only the number of page faults written)

i) FIFO replacement (10 faults)



9	9	9	9	9	8	8	8	8	8	0	0
–	5	5	5	5	5	2	2	2	2	2	7
–	–	3	3	3	3	3	1	1	1	1	1
–	–	–	6	6	6	6	6	9	9	9	9

ii) Optimal replacement (09 faults)

9	9	9	9	9	9	9	9	9	9	0	7
-	5	5	5	5	8	2	1	1	1	1	1
-	-	3	3	3	3	3	3	3	3	3	3
-	-		6	6	6	6	6	6	6	6	6

Q5 a) At Line X: value = 10 (Global variable value)
At Line Y: value = 30 (Local copy created within the program)

2+2 marks

b) External fragmentation - As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous, that is, storage is fragmented into a large number of small holes. If all these small pieces of memory were in one big free block instead, we might be able to run several more processes.

No, Paging does not suffer from external fragmentation.

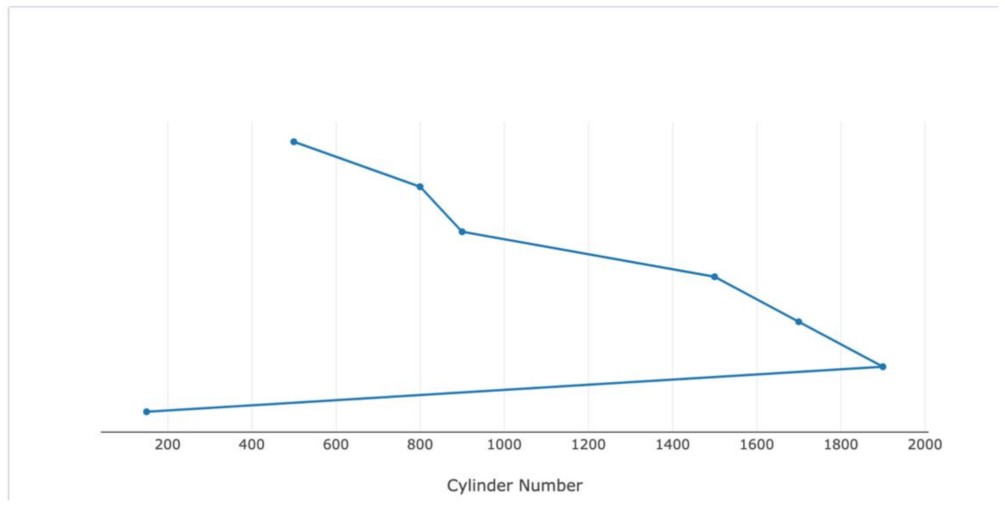
Reason: Paging divides memory into fixed-size pages, ensuring memory allocation in uniform units. This eliminates the problem of external fragmentation.

(2+1+2)

c) 150 1500 900 1700 800 1900
Initial position: 500

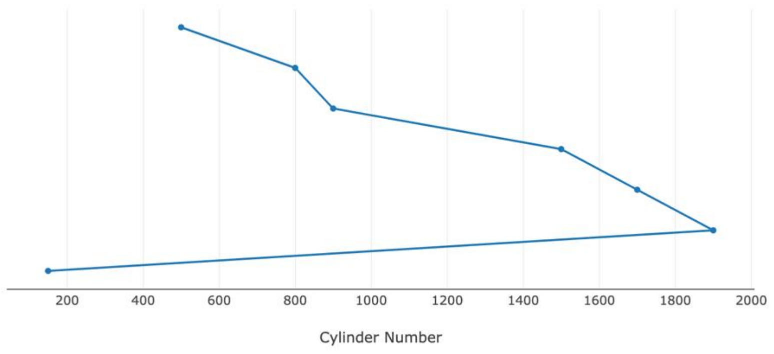
3 + 3 marks

Shortest Seek Time First
500 – 800 – 900 – 1500 – 1700 – 1900 – 150



Distance=300 + 100 + 600 + 200 + 200 + 1750

LOOK Scheduling
500 – 800 – 900 – 1500 – 1700 – 1900 – 150



Distance=300 + 100 + 600 + 200 + 200 + 1750

Q6 a)

Segment	Base	Length
0	519	500
1	1800	95

(1 marks each)

2	170	300
3	1920	780
4	1860	50

Physical addresses:

i) 0, 240 = 519 + 240 = 759
ii) 2, 320 = Invalid as 320 > Length
iii) 3, 670 = 1920 + 670 = 2590
iv) 4, 10 = 1860 + 10 = 1870

b) Translation Look aside Buffer (TLB) hit ratio = 80% (2+3)
Memory access time (ma) = 150 ns
TLB buffer access time = 10 ns:

i) Effective memory access time without TLB
EMAT = 2 X 150 = 300

ii) Effective memory access time with TLB
EMAT = 0.80(10+150)+0.20(10+150+150)
= 128 + 62 = 190

c) (2 X 3 marks)

i) TLB miss with no page fault
Situation is the page is in memory but TLB is not updated.
Frame number is retrieved by using Page number as an index in the page table

ii) TLB miss and page fault
Situation is page not brought in memory hence page fault occurred
Frame: Trap to the OS as it is a page fault

iii) TLB hit
Situation is page is in memory and TLB is updated to include the page
Frame is retrieved from the TLB

Q7 a) Page size = 2 KB = 2048 bytes (2+2 marks)

[IF THE STUDENT HAS DONE USING BINARY METHOD, PLEASE GIVE MARKS]

i) 3085
Page Number = 3085 / 2048 = 1
Offset = 3085 % 2048 = 1037

ii) 42095

		Page Number = 42095 / 2048 = 20 Offset = 42095 % 2048 = 1135	
	b)	Belady’s anomaly: For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases. i) FIFO: Suffers from Belady’s anomaly ii) Least Recently Used algorithm (LRU): Not suffers from Belady’s anomaly	(3+2)
	c)	Available partitions: 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, 125 KB (in order) Process to schedule: 115 KB, 500 KB, 375 KB (in order) i) First-fit algorithm 115 KB => 300 KB (185 KB remaining) 500 KB => 600 KB (100 KB remaining) 375 KB => 750 KB (375 KB remaining) ii) Best-fit algorithm 115 KB => 125 KB (KB remaining) 500 KB => 600 KB (100 KB remaining) 375 KB => 750 KB (375 KB remaining)	(6)