Typical answer – Operating System 1

Questions

1. The offset in a logical address can be greater than the offset in a physical address. True or false? Explain.

False.
The offset; whether in logical or physical address takes its values within the range of the size of page. The size of a page equals the size of a frame. Accordingly, the value of an offset in logical address can never be greater or less than the offset in physical address. **0.5**

2. The page table is unique in an OS. True or false? Explain.

False.
The page table is specific for each process. Hence, at time t, the number of the page tables in an OS depends of the number of processes at that time. **1**

3. The process table is unique in an OS. True or false? Explain.

True.
The process table is an OS data structure. It gathers all needed management information for the processes in an OS. **0.5**

4. How to define a starvation (famine) in OS?

The starvation (or famine) in OS is the deprivation of a process from getting benefit of some OS resources. The deprivation is due to a bad management algorithm for the resource. **0.5**

Give examples for:

(a) CPU, and

SJF or SRT scheduling can lead to CPU process famine. **0.5**

(b) Memory.

The multi-programming with fixed partition with unique queue can lead to memory famine. **0.5**

5. The size of logical memory can be greater than physical memory. True or false? Explain.

True.
Te main idea of the memory paging (logical memory) is indeed to free the OS from the memory space obstacle. In paging, a process with a size that is greater than the available physical size is possible. **1**

6. In a single-processor computer, tell which process state possibilities are true and which are false in the table below.

#	Ready	Running	Waiting
1	1	0	0
2	0	1	0
3	0	0	4
4	2	1	0
5	3	0	3
6	0	2	1
7	3	3	3
8	0	2	0

1-False, 2-True, 3-True, 4-True, 5-False, 6-False, 7-False, 8-False **0.25 each**

Exercise 1. (7 points) Consider the execution of the following processes/jobs on a single-processor machine.
NB. A small value of priority indicates a higher priority.
1. considering only the non-grayed part of the table, give the process/job execution diagrams using:
(a) the SRT,
(b) RR (quantum=1) scheduling, and
(c) the preemptive priority scheduling algorithm.

2. Now we consider the whole data of table, draw the execution diagram for a RR scheduling (Quantum=2)?

See the three figures in the attached file bellow.

3. For the last case, calculate the following parameters for each process/job:

(a) CPU usage,

CPU usage = CPU time / RT **0.5**

CPUU(A) = 5/15.000.1 %
CPUU(B) = 4/15 %
Etc.

(b) Response time, and
RT = EET – Arrival date **0.5**

RT(A) = 15.0001 - 0
RT(B) = 17.0001 – 2.0001
RT(C) = 12.0001 – 2.0001
RT(D) = 20.0001 – 4.0001
RT(E) = 14.0001 - 3

(c) Waiting time.

WT = RT – CPU time **0.5**

WT(A) = 15.0001 - 5
WT(B) = 15 - 4
WT(C) = 10 - 2
WT(D) = 16 - 4
WT(E) = 11.0001 – 3

Exercise 2. (7 points) Let the main memory of a computer be 2^{10} address units. We assume that this memory is managed in multiprogramming in 5 fixed partitions with the k counter technique.
The partitions have the following sizes: 50, 200, 40, 300, 150 memory words. The rest of the memory is occupied by the OS. Let consider also k=2.
At first, partitions are occupied; they are freed at times 4, 0.2, 4.1, 0.5, 2. The processes come to memory following the table below.

Process	Size	Arrival date	Stay time in memory
A	20	2.3	1
B	120	3.5	2
C	35	0	0.6
D	110	4	2
E	40	2.1	4.5
F	290	1	4

1. Draw the memory diagram, knowing that the OS occupies the lower addresses.

The memory size = 2^{10} address unit (memory word). The partitions occupy a size = 50+200+40+300+150=740 word.
The OS size = 2^{10} – 740 = 1024 – 740 = 284 word. **0.5**

150
300
40
200
50
OS

2. What would be the content of the queue at time 2.4?

To know the content of the queue at time 2.4, we have to know the content of the partitions in all times before that moment.
The diagram bellow depicts the content of the partitions at every important moment in memory operations. **3**

150	150	150	150	150	150	B, 30	B, 30	B, 30	150
300	300	F, 10	F, 10	F,10	F,10	F, 10	F, 10	300	300
C, 5	40	40	40	A, 20	40	40	40	40	40
200	200	200	200	200	200	200	D, 90	D, 90	D, 90
50	50	50	E, 10	E,10	E,10	E, 10	E, 10	E, 10	E, 10
OS	OS	OS	OS	OS	OS	OS	OS	OS	OS
t=0 access of C	t=0.6 exit of C	t=1 access of F	t=2.1 access of E	t=2.3 access of A	t=3.3 exit of A	t=3.5 access of B	t=4 access of D	t=5 exit of F	t=5.5 exit of B

150	150
300	300
40	40
D, 90	200
50	50
OS	OS
t=6 exit of D	t=6.6 exit of E

According to the diagram above, the queue is empty at time 2.4. It is the time between 2.3 and 3.3.

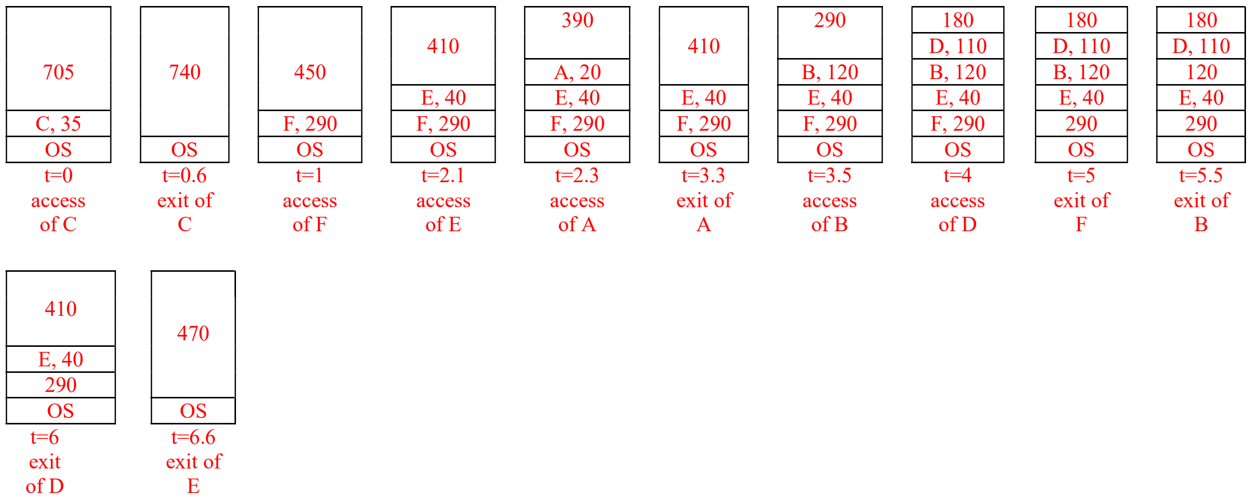
3. What is the total amount of internal fragmentation at time 6?

According to the diagram above and exactly at t=6, the total amount of internal fragmentation is 90.
1

If we manage this memory using variable partitions and the Worst Fit placement strategy.

4. Provide a diagram showing the contents of the memory at time 4?

To know the content of the memory at time 4, we must draw the diagrams that depict all important moments. See the diagram bellow. **1.5**



At time 4, the content of the memory is depicted in the subfigure N° 8 (from left).

5. What is the amount of external fragmentation at time 4.2?

180 **0.5**

6. Is there internal fragmentation? If so, how size is it?

No. In the multi-programming with variable partitions, there is no internal fragmentation. **0.5**