1.

a.

Answer) The TLB is a small, fast-lookup hardware cache that stores a few entries from the page table. It improves memory access times by allowing the memory management unit (MMU) to check the TLB first. If the page number is found in the TLB (TLB hit), the frame number is available immediately, avoiding the need for two memory accesses.

b.

Answer: Memory access slows down by a factor of 2 because the system first needs to access memory to retrieve the page frame number from the page table, and then it has to access memory again using the physical address (page frame number + offset).

c.

Answer) The main purpose of the multilevel page table method is to avoid keeping all the page tables in memory at once, reducing the memory overhead associated with large page tables.

d.

Answer) A hashed page table aims to efficiently manage larger address spaces by reducing the number of entries in the page table and minimizing memory overhead.

e.

Answer) The number of page table entries in the inverted page table is equal to the number of page frames in physical memory.

2.

a.

Sol) Possible maximum size of virtual space = $2^{32} = 2^{22} \times 2^{10} = 2^{22}$ **KB**

$\therefore$ Maximum # of pages per a process = virtual space / a page size = $2^{22}/2 = 2^{21}$ pages .

Since maximum number of pages is 2^{21} pages, there are 2^{21} possible entries in a page table.
Maximum size of page table per a process = number of pages $\times$ one entry size

$= 2^{21} \times 64$ bits $= (2^{21} \times 64)/8$ Byte $= 2^{21} \times 8$ byte= **16 MB**

b.

Sol) simply need calculate the number of page frames

# of page frame = size of RAM / size of page = 16GB / 2KB = $16 \times 2^{30} / 2 \times 2^{10} = 2^{23}$ page frames $\therefore$ **23 bits for page frame number.**

3.

a)

$1 - p^n$

b)

n= (10 GB – 2 GB)/1GB = 8

p= 0.85 CPU utilization =1- $(0.85)^8$ = 0.7275 about 73%

c)

n = (20 GB – 2GB) / 1GB = 18, p = 0.85 $1 – (0.9)^{18}$ = 0.8499 about 85 %

4.

a. 2^{14} = 16 KB

b. 2^{30} page tables +1

c. 2^{30} x 2^{20} pages = 2^{50} pages

d. There are 16GB /16 KB = $16 \times 2^{30} / 16 \times 2^{10} = 2^{20}$ page frames. The system need reserve 20 bit for page frame number.

5.

Shortest remaining time first (preemptive):

P_1	P_2	P_3	P_5	P_2	P_4	P_1
3	5	9	12	17	24	32

- Average waiting time – ((24-3)+(12-5)+0+(17-7)+(9-8))/5 =(21 + 7 + 0 + 10 + 1)/5 = 7.8
- Average turnaround time – ((32-0)+(17-3)+(9-5)+(24-7)+(12-8)/5= (32 + 14 + 4+ 17 + 4)/5 = 14.2

Preemptive priority queue:

P1	P2	P3	P4	P2	P1	P5
3	5	9	16	21	29	32

- Average waiting time – (18+11+ 0+ 2+ 21)/5 =10.4
- Average turnaround time – (29+ 18+ 4+ 9+ 24)/5 = 16.8

6.

Sol) Let's assume the following values at time t: mutex = 1, full = 0, and the CPU scheduler selects the consumer.

- The consumer performs a down operation on mutex (making mutex = 0).
- The consumer then attempts a down operation on full. Since full = 0, the consumer goes to sleep, waiting in the queue for full.
- The CPU scheduler selects the producer.
- The producer performs a down operation on empty (making empty = N - 1).
- The producer then attempts a down operation on mutex. Since mutex = 0, the producer also goes to sleep, waiting in the queue for mutex.
- At this point, both the consumer and the producer are asleep, leading to a deadlock.

7.

- Running state – a process running on CPU
- Ready state – a process waiting for CPU
- Blocked state – a process waiting for I/O finish
- Transaction 1 – a process need I/O
- Transaction 2 – a process time out
- Transaction 3 – CPU scheduler select a process to run
- Transaction 4 – a process finish I/O and ready to run

8.

Sol)

- Bitmap: #of allocation unit = 2GB/4KB = $(2 \times 2^{30})/(4 \times 2^{10}) = 2^{31}/2^{12} = 2^{19}$ units
  Size of the bitmap = 2^{19} bits = **2^{16} byte**

- The linked list: number of node for linked list= 2 GB /64KB = $2^{31}/2^{16}$ or 2^{15} nodes.
  size of each node = 32+16+16 = 64 bit = 8 byte =2^3 bytes
  Total size of linked list = number of node $\times$ size of a node = $2^{15} \times 2^3$ bytes = **2^{18} bytes**.

9.

a)

virtual address $19845/(4 \times 2^{10}) = 4.84$ in page #4 (map to page frame #135).
virtual page #4 begin with address $4 \times 4 \times 2^{10}$
page frame #135 is start with address $135 \times 4 \times 2^{10}$
physical address = $135 \times 4 \times 2^{10} + (19845 - 4 \times 4 \times 2^{10}) = 552{,}960 + 3{,}461 = 556{,}421$

b)

virtual address $21581/(4 \times 2^{10}) = 5.26$ in page #5
virtual page #5 begin with address $5 \times 4 \times 2^{10}$
page frame #95 is start with address $95 \times 4 \times 2^{10}$
physical address = $95 \times 4 \times 2^{10} + (21581 - 5 \times 4 \times 2^{10}) = 389120 + 1101 = 390221$

10.

sol)
let's assume Permit =0 at time T
P_0 tries to enter C.S. and can enter since Permit =0.
P_0 finish its job in C.S. and set Permit =1
P_1 is currently running outside C.S ,it is terminated with fatal error.
P_0 tries to enter C.S. again but P_0 never can.

11.

Sol) If each job has 50% I/O wait, then it will take 40 minutes to complete in the absence of competition. If run sequentially, the second one will finish 80 minutes after the first one starts. With two jobs, the approximate CPU utilization is $1 - 0.5^2 = 0.75$. Thus, each one gets 0.375 CPU minute per minute of real time. To accumulate 20 minutes of CPU time, a job must run for20/0.75 +20/0.75 minutes, or about 53.33 minutes. Thus running sequentially the jobs finish after 80 minutes, but running in parallel they finish after 53.33 minutes.