1.

a.

1 block = $2 \times 2^{10} \times 8$ = 16384 bit, 16384 /32 = 512 -1 = 511 block numbers /block
128 GB = $(128 \times 2^{30})/(2 \times 2^{10}) = 2^{26}$ blocks (number of blocks in 128 GB)
Needs (2^{26}) /511 = 131328.5= 131229 blocks

b.

128 GB = $(128 \times 2^{30})/(2 \times 2^{10}) = 2^{26}$ blocks (number of blocks in 128 GB)
System need one bit per block, the bit map size is 2^{26} bits.
2^{26} bits = $(2^{26})/ 8 = 2^{23}$Byte
Since each block size is 2KB, need $(2^{23})/ (2 \times 2^{10})$= 4096 blocks to save free block information.

c.

- Since this system use 32bit disk block number, this system support 2^{32} blocks
- Maximum disk size = $2^{32} \times 2 \times 2^{10}$ Byte =8×2^{40} = 8 TB

2.

a) files are cached in the RAM when it is opened.

b)

- In LSF, each i-node is not at a fixed location; they are written to the log.
- LFS uses a data structure called an **i-node map** to maintain the current location of each i-node.
- Opening a file consists of using the map to locate the i-node for the file.

3.

a)

- Maximum virtual address space = $2^{32} = 2^{22} \times 2^{10} = 2^{22}$ **KB**
- ∴Maximum # of pages per a process = virtual space / a page size = $2^{22}/4 = 2^{20}$ pages .
- Maximum size of page table per a process = number of page × one entry size
  = $2^{20} \times 64$ bits = $(2^{20} \times 64)/8$ Byte= $2^{20} \times 8$ byte= **8 MB**

b)

- need calculate the number of page frame
  # of page frame = size of RAM / size of page
  = 8GB / 4KB = $8 \times 2^{30} / 4 \times 2^{10} = 2^{21}$ page frames
  ∴ **21 bits for page frame number.**

4.

Sol) In LINUX system, size of a block and number of blocks information are saved in super block in the partition. Since used block information for a file is saved in it's i-node, we can get used blocks information by scanning all i-nodes.

```
Create new bitmap (size of bit = number of block from super block)
        Reset (set as 0)
For each i-node do
        For each entry of i-node do
                Set bitmap to 1 (based on used block information)
```

5.

a)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	2	3	0
P_1	3	0	2	0	2	0			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

A = (2, 3, 0) – P_1 – (5, 3, 2) – P_3 – (7, 4, 3) –P_0 –(7, 5, 3)- P_2 –(10, 5, 5) –P_4 –(10, 5, 7)

b)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	4	3	0	1	2
P_1	2	0	0	1	2	2			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	3	2	2	1	1	1			

A = (0, 1, 2) – P_3 – (2, 2, 3) – P_1 – (4, 2, 3) –P_4 –(7, 4, 5)- P_0 –(7, 4, 5) –P_2 –(10, 5, 7)

c)

Process	C			R			A		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	2	3	0	0	2
P_1	2	0	0	1	2	2			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	3	3	2	1	0	1			

Answer) non- of process can

6.

a) Page 2
b) Page 0
c) Page 1
d) Page 0

7.

a.

$$Total\ Overhead(P) = Average\ page\ table\ size + the\ wasted\ memory\ in\ th\ last\ page\ of\ process$$
$$= \frac{S}{P} \times E + \frac{P}{2}$$

b.

$$Overhead'(P) = -\frac{SE}{P^2} + \frac{1}{2} = 0$$

$$P = \sqrt{2SE} : optimal \text{ page size}$$

8.

a)

- Mutual exclusion
- Hold-and Wait
- No preemption
- Circular wait

b)

- Ignore
- Detection and recovery
- Avoidance with dynamic allocation
- By attacking one of necessary deadlock condition

c) a segment is a logical entity.

- If the segments are large, to keep them in the physical memory might be wasting memory space.
- If a segment's virtual space is larger than physical space, it is not even possible to keep them in the physical memory.

9.

Sol) since 1 block is 2KB, and 16 Byte per block address, it can save $2 \times 2^{10} / 16 = 2^{11}/2^4 = 2^7$ = 128 block information

Total = 128 +8 = 136 block information.

Since a block size is 2KB, largest file will be 2KB × 136 =272 KB

10.

a)

P_3

A=(2,1,0,0) -> (2,2,2,0) Deadlock

b)

P_1 P_3 P_2 P_4 P_5

$A=(0,0,0) \rightarrow (0,1,0) \rightarrow (3,1,3) \rightarrow (5,1,3) \rightarrow (7,2,4) \rightarrow (7,2,6)$

11.

Since P_1 need 3 R_5 in total minimum Y should be >=2.
Since R_1=0, R_2 =0, only P_4 can be selected based on A with X >=1

- with X=1, Y=2 A=(0 0 1 1 2)
  after P_4, A = (0 0 1 1 2) + (1 1 1 1 0) = (1 1 2 2 2)
  after P_1, A = (1 1 2 2 2) + (1 0 2 1 1) = (2 1 4 3 3)
  after P_3, A = (2 1 4 3 3) + (1 1 0 1 0) = (3 2 4 4 3)
  after P2, A = (3 2 4 4 3) + (2 0 1 1 0) = (5, 2 5 5 3)

12.

Solution 1)

Sol)
Attacking hold and wait,
starvation

Solution 2)

Sol)
Attacking circular wait,
If a process need two resource at a same time, this solution have problem