1.

a. All Coffman conditions are true.
   - Mutual Exclusion: true. Resource Y has 1 instance and that instance can be used by only 1 thread at a time. It is non-sharable.
   - Hold and Wait: true. Thread 0 is holding resource Y and also waiting for an instance of resource X which is held by thread 1.
   - No Preemption: true. There is no indication that an outside force can release a resource from a pthread holding it. Only the thread owning that resource can voluntarily release the resource.
   - Circular Wait: true. Thread 0 is waiting for an instance of resource X which is held by thread 1 and thread 2. Thread 2 is waiting for resource Y which is held by thread 0. The circular wait exists between thread 0 and thread 2.

b. There is no deadlock in the current state. Thread 0 is waiting for only 1 instance of resource X. An instance of resource X is being held by thread 1. Thread 1 can execute, terminate on its own, and release the instance of resource X. Next, thread 0 can acquire it, execute, finish, and release resource Y. Then thread 2 can finally acquire resource Y can do its job. At the end, no thread is stuck forever.

2.

a. and c.

For a segment table, there is no length regulation for each segment in the physical memory. If we allow each segment to grow and shrink without a limit, they will overlap each other and corrupt memory. Therefore, beside a segment base which indicates where the segment starts from memory, each entry in a segment table needs a limit of that segment to show where does this segment ends. This limit is used to check if a logical address is valid or not. It is valid if its offset from the base is smaller than the limit, which means that logical address is still inside the segment.

On the other hand, a page table does not need limit, but only has frame number because each frame and each page have fixed size, and all of them have the same size. This means that no logical address can be outside the valid range, since page (logical address) has the same size with frame (physical address). The page table only needs a frame number to map each unique page to each unique frame.

b. We can definitely record base in a page table. The reason we do not do it is frame number can do that job well enough. Because every frame has the same size, the computer just needs to calculate frame size * frame number to find the physical address and access the memory from there. Also, the frame method helps clear separation between the view of logical memory and actual physical memory for programmers. One only needs to to take care of logical address while coding.

3.

a. 32 - 47, 64 - 95, 112 - 127

b.

Logical Address	Physical Addresss
1	33
50	18
95	Impossible (page fault)
96	48
120	Impossible (page fault)

4.

a. FIFO

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5
Frame 0	7	7	7	1	1	1	1	1	6	6	6	6	0	0
Frame 1	X	2	2	2	2	5	5	5	5	7	7	7	7	5
Frame 2	X	X	3	3	3	3	3	4	4	4	4	1	1	1
Page fault? (Y/N)	Y	Y	Y	Y	N	Y	N	Y	Y	Y	N	Y	Y	Y

Total page faults = 11

b. LRU

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Reference	7	2	3	1	2	5	3	4	6	7	7	1	0	5
Frame 0	7	7	7	1	1	1	3	3	3	7	7	7	7	5
Frame 1	X	2	2	2	2	2	2	4	4	4	4	1	1	1
Frame 2	X	X	3	3	3	5	5	5	6	6	6	6	0	0
Page fault? (Y/N)	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y

Total page faults = 12