Question 1: What would you do if you were a cache controller and received load/store from the CPU addressing word, W, in block B

If B is in your cache (valid=1 and the tags match)
- if *Load (i.e. a lw instruction)*
  - return W to CPU
- if *Store (i.e. a sw instruction)*
  - write W
  - if B is SHARED
    - send "invalidate B" on bus
    - change B's state to EXCLUSIVE

If B is not in your cache (either valid = 0 or tags do not match)
- if you have to evict an EXCLUSIVE block to make room for B, then write the evicted block back.
- if *Load*, put a "request B to read" on the bus
- if *Store*, put a "request B to write" on the bus
- wait for bock B to appear on the bus (sent by memory or by another cache)
- if *Load*, return W to CPU and set B's state to SHARED
- if *Store*, write W and set B's state to EXCLUSIVE

Question 2: What would you do if you were a snooping module and detect a message on the bus involving block B?

Look for block B in your cache (check valid bit and compare stored tag with B's tag)
If B is not in your cache (valid bit = 0 and tags do not match),
- do nothing

Else:
- If the message is "invalidate B"
  - set B's valid bit to 0
- If the message is "request B to read"
  - if B is SHARED (dirty bit = 0), do nothing
  - if B is EXCLUSIVE (dirty bit = 1), write B back and change its state to SHARED
- If the message is "request B to write"
  - if B is SHARED (dirty bit = 0), set B's valid bit to 0
  - if B is EXCLUSIVE (dirty bit = 1), write B back and set B's valid bit to 0

Question 3: 7

Consider the snooping cache coherence protocol for a bus-based system with three processors P1, P2, and P3, each with a direct-mapped cache and assume that the size of a cache line (block) is one word. The following sequence of operations access two memory locations (words), A and B, that are mapped to the same cache location. Initially A = B = 8 (in memory) and they are not cached in any cache. For each memory access, determine the content and state of the cache line in each processor and the bus operations resulting from the coherence protocol

P_1 writes B=5
P_3 reads B
P_2 reads A
P_3 writes B=0
P_3 writes B=12
P_2 writes A=10
P_1 reads A
P_3 reads A
P_2 writes A = 12

Event	P1's cache	P2's cache	P3's cache
P1 writes B=5 Write miss	Request B to write L ← B=5 (E)	L ← (I)	L ← (I)
P3 Reads B Read miss	Write back B L ← B=5 (S)	L ← (I)	Request B to read L ← B = 5 (S)
P2 reads A Read miss	L ← B=5 (S)	Request A to read L ← A=8 (S)	L ← B = 5 (S)
P3 writes B=0 Write hit	L ← (I)	L ← A=8 (S)	Send invalidate B L ← B = 0 (E)
P3 writes B=12 Write hit	L ← (I)	L ← A=8 (S)	L ← B = 12 (E)
P2 writes A=10 Write hit	L ← (I)	Send invalidate A L ← A=10 (E)	L ← B = 12 (E)
P1 reads A Read miss	Request A to read L ← A=10 (S)	Write back A L ← A=10 (S)	L ← B = 12 (E)
P3 Reads A Read miss	L ← A=10 (S)	L ← A=10 (S)	Write back B Request A to read L ← A = 10 (S)
P2 writes A=12 Write hit	L ← (I)	Send invalidate A L ← A=12 (E)	L ← (I)

Question 4: 7

Repeat question 3 assuming that A and B are in the same cache block, X, mapped to location L (Initially, A=B=8 in Memory)

P_1 writes B=5
P_3 reads B
P_2 reads A
P_3 writes B=0
P_3 writes B=12
P_2 writes A=10
P_1 reads A

Event	P1's cache	P2's cache	P3's cache	In Memory
P1 writes B=5 Write miss	Request X to write L ← A=8, B=5 (E)	L ← (I)	L ← (I)	memory supplies X A=8, B=8
P3 Reads B Read miss	Write back X L ← A=8, B=5 (S)	L ← (I)	Request X to read L ← A=8, B = 5 (S)	A=8, B=5
P2 reads A Read miss	L ← A=8, B=5 (S)	Request X to read L ← A=8, B=5 (S)	L ← A=8, B = 5 (S)	memory supplies X A=8, B=5
P3 writes B=0 Write hit	L ← (I)	L ← (I)	Send invalidate X L ← A=8, B = 0 (E)	A=8, B=5
P3 writes B=12 Write hit	L ← (I)	L ← (I)	L ← A=8, B = 12 (E)	A=8, B=5
P2 writes A=10 Write miss	L ← (I)	Request X to write L ← A=10, B=12 (E)	Write back X L ← (I)	A=8, B=12
P1 reads A Read miss	Request X to read L ← A=10, B12 (S)	Write back X L ← A=10, B=12 (S)	L ← (I)	A=10, B=12