Recall: ToyOS has a request handler (ioReqHndlr) and possibly an interrupt handler (ioIntHndlr) for each io device. These handlers make use of functions updateRunqPcb, scheduler, wakeup, wait, among others. OS-IO interaction can have various features: program-driven, interrupt-driven, synchronous, asynchronous, dma, etc.

1. [20 points]
Consider an output device $X$ as follows: (1) data block size is 2 memory words (e.g., 64 bits); (2) time to output a data block (i.e., from adaptor to device) is the time that the cpu takes to execute about 10 machine instructions.

Part a. Which features would you choose for OS–$X$ interaction. Explain briefly.

Solution [8 pt]

- Program-driven vs interrupt-driven: program-driven
  Because data block size and output time are small, interrupt handling would be less efficient. [1 pt]
  [2 pt]
- Synchronous vs asynchronous (i.e., whether user blocks until io completed): either is ok
  Because the output time is so small, synchronous makes more sense.
  Asynchronous makes sense only if many (e.g., hundreds) requests are issued simultaneously (by different threads). [1 pt]
  [2 pt]
- No-dma vs dma: either is ok
  Actually, dma does not make sense because the benefit is minor. [1 pt]

Part b. Write down the request handler and the interrupt handler (if any) for $X$, consistent with your part a answer. You should use a similar level of detail as in ToyOS.

Solution [12 pt]

For program-driven, synch, no dma:

- $x$ReqHndlr(addr)
  // intrpts off, kernel
  while (x.ctrl.busy) skip // busy wait // output code
  write mem[addr, addr+1] to x.dbr // output code
  set x.ctrl to start io // output code
  rti [8 pt]
- No interrupt handler [4 pt]

If we had chosen intrpt-asynch in part a:

- kernel thread, say $x$Server, interacts with $X$
- $x$reqQ: request queue
  xQ: pcb queue; xServer thread waits here
- $x$ReqHndlr(addr)
  wakeup(xQ) // suff to do this only if xQ is empty
  add req to reqQ
  rti [8 pt]
- $x$IntHndlr()
  wakeup(xQ) [4 pt]

Other choices in part a would be “in between” the above two.
2. [20 points] Repeat problem 1 but with device \( X \) replaced by a keyboard. (So your answer has part a and part b.)

Part a.

**Solution [12 pt]**

- Program-driven vs interrupt-driven: interrupt-driven
  - Input is externally initiated and slow, so program-driven would be horribly inefficient.

- Synchronous vs asynchronous: synchronous
  - Unlikely that user can proceed without keyboard input.

- No-dma vs dma: no-dma

Part b.

**Solution [8 pt]**

interrupt, synch, no dma:

- \( xQ \): pcb queue, process at head gets next input

- \( xReqHndlr(\text{addr}) \)
  - // intrpts off, kernel
  - wait(\( xQ \))
  - read input from keyboard dbr
  - re-enable interrupt at keyboard ctrl
  - rti

- \( xIntHndlr(\text{addr}) \)
  - // intrpts off, kernel
  - wakeup(\( xQ \))
  - rti