```
1. In an ARM system supporting IRQ interrupts, e.g. KBD interrupts,
   the following components are needed/provided:
(1). Vector tabble at memory address 0
     0x18: LDR PC, irq_handler_addr
     irq_handler_addr: .word irq_hanler

(2). irq_handler:
       sub lr, lr, #4
       stmfd sp!, {r0-r12, lr}
       bl  IRQ_handler
       ldmfd sp!, {r0-r12, pc}^

(3). IRQ_handler{
       if (VIC.statusRegBit31 && SIC.statusRefBit3)
           kbd_handler();
     }
   int hasData = 0;
   char c;

(4). kbd_handler()
     {
       int hasData;
       get scancode;
       c = ASCII char mapped by scancode;
       hasData = 1;
     }

(5). char kgetc()
     {
        while(hasData==0);
        lock()
          hasData = 0;
        unlock();
        return c;
     }

(6). mainProgram()
     {
        unlock();    // allow CPU to accept IRQ interrupts
        kgetc();     // CPU executed this
     }
Assume: the CPU has executed kgetc() in mainProgram().

1. Draw a diagram to show the control flow of CPU when a key is pressed at KBD
```

kbd → interrupt handler → char buf[N], int data, head, tail → getc() → process
|lower-half| |input buffer, as well as control vars| |upper half; destroys buffer|

2.in (2): what does the line

```
ldmfd sp!, {r0-r12, pc}^
```

do? **Returns the registers r0-r12 and program counter back to stack**

3. How to tell it's a KBD interrupt? **Check if SIC bit == 3.**

4. in (5), WHY are the lock(); .. unlock() needed? **Lock() signals that there is an IRQ interrupt being processed. While unlock() lets the CPU know it can accept IRQ interrupts again.**