(20) Question 1. Find the sequence of execution that leads to the incorrect value being displayed.
Part a) If `Time` is any value with a '9' in the ones digit (e.g., 49), and the interrupt occurs between the two `SCI_OutChar`, then the display will be wrong (the time is either 49 or 50 but the display shows 40).

```
1 SCI_OutChar(Time/10+0x30); // tens
5 SCI_OutChar(Time%10+0x30); // ones
}
```

```
2 Time++;   // 0 to 59
3 if(Time == 60) time=0;
4 RTIFLG = 0x80;
```

Part b) One way to fix this bug is to read time once and display the copy (my favorite)

```
void Display(void){ unsigned short myTime=Time;
 SCI_OutChar(myTime/10+0x30);       // tens digit
 SCI_OutChar(myTime%10+0x30); }     // ones digit
```

Another way to fix this bug is to disable interrupts during the display (worse latency)

```
void Display(void){ unsigned char saveCCR; asm("tpa\n  staa %SaveCCR\n  sei"); // atomic
 SCI_OutChar(Time/10+0x30);        // tens digit
 SCI_OutChar(Time%10+0x30);        // ones digit
asm("ldaa %SaveCCR\n tap");}       // restore old interrupt status
```

(50) Question 2. Part a) Prototypes for public functions goes in the `IO.h` header file.

```
void IO_Init(void); // initialize, call this function once before Input/Output
unsigned char IO_Input(void); // wait for key to be pressed, return ASCII code
void IO_Output(unsigned char); // output ASCII code on display
```

Part b) The functions themselves go in the `IO.c` implementation file. Notice that START DONE and RW are private.

```
#define START 0x80
#define DONE 0x40
#define RW 0x20
void IO_Init(void){ // initialize, call this function once before Input/Output
  DDRA = (DDRA&0x1F)|0xA0;   // output PA7=start, input PA6=done, output PA5=R/W
  DDRB = 0;}                 // default case as input
void IO_Output(unsigned char data){ // output ASCII code on display
  PORTA &= ~RW;              // 1) set R/W to 0
  DDRB = 0xFF;               // 2) make Port B outputs
  PORTB = data;              // 3) write Port B with the 8-bit data to display
  PORTA |= START;            // 4) set start high
  while((PORTA&DONE)==0){}; // 5) wait for done to be 1
  PORTA &= ~START;           // 6) set start low
  while(PORTA&DONE){};       // 7) wait for done to be 0
  DDRB = 0;}                 // 8) make Port B inputs again
unsigned char IO_Input(void){ // wait for key to be pressed, return ASCII code
unsigned char data;
  PORTA |= RW;               // 1) set R/W to 1
  PORTA |= START;            // 2) set start high
  while((PORTA&DONE)==0){}; // 3) wait for done to be 1
  data = PORTB;              // 4) read input data
  PORTA &= ~START;           // 5) set start low
  while(PORTA&DONE){};       // 6) wait for done to be 0
  return data;}
```

(30) Question 3. Since `i2` is static, it exists in global space (seen as `L2`), and not on the stack.