a. For an FOSC = 40 MHz, what value must be written to the SBPRG register to achieve a baud rate of 9600 in high speed mode?

\[
SBPRG = \frac{\text{FOSC}}{(\text{baud} \times k)} - 1 = \frac{(40e6)}{(9600 \times 16)} - 1 = 259.4 = 259.
\]

This value > 255, so this baud rate is unreachable in high speed mode. Would have to use low speed mode (K = 64), then SBPRG = \[(40e6)/(9600*64)] – 1 = 64.

b. Write C code that waits for a character to become available at the serial port and returns that character (i.e., write the getch() function).

```
getch() {
    while (!RCIF) ;  // wait for character
    return(RCREG);
}
```

c. Write a C code fragment that turns an LED connected to port RB3 (high true LED) if an overrun error has occurred in the serial port. Identify the BIT number and REGISTER that the overrun error is in.

```
if (OERR) {       //OERR bit is RCSTA reg, bit #1
    RB3 =1 ;
}
```

d. What is the problem with sending a large number of bits at one time in asynchronous serial IO? Why are we restricted to only a few bits (< 10)?

The sender and receiver do not share common clocks; the clocks are not perfectly matched so clock mismatch in bit sampling accumulates for each bit sent. If enough bits are sent, the receiver will not sample the input line in the correct place when the cumulative clock mismatch becomes greater than ½ of a bit time.

e. In the diagram below, write C code that turns the LED off, then goes into an infinite loop in which the LED is toggled for each press and release of the switch (toggle means that if the LED is on, then turn off; if it is off, then turn it on). Assume the port is configured already.

```
RB0 = 0;
while(1) {
    while(1) {
        while (RB1);  // wait for press/release
        RB0 = 1;      // turn on
        while (RB1);
        while (!RB1); // wait for press/release
        RB0 = 0;      // turn off
    }
}
```