PreLab 1

1.

a. The following are lines of code that determine which UART to activate. You must also activate the GPIO port corresponding to that UART.

```
SYSCTL_RCGCUART_R |= 0x01;          // activate UART0
SYSCTL_RCGCGPIO_R |= 0x01;          // activate port A
```

b. The following determines the baud rate, parity, data bits, and stop bits.

```
#define UART_LCRH_WLEN_8       0x00000060  // 8 bit word length
#define UART_LCRH_FEN          0x00000010  // UART Enable FIFOs

UART0_IBRD_R = 27;                  // IBRD = int(50,000,000 / (16 * 115,200)) =
int(27.1267)
UART0_FBRD_R = 8;                   // FBRD = int(0.1267 * 64 + 0.5) = 8
                                // 8 bit word length (no parity bits, one stop bit, FIFOs)
UART0_LCRH_R = (UART_LCRH_WLEN_8|UART_LCRH_FEN);
```

c. The port pins vary by UART used. In this example UART0 uses PA0 to receive and PA1 to transmit.

d. The following code inputs one byte from the UART port.

```
// input ASCII character from UART
// spin if RxFifo is empty
char UART_InChar(void){
  char letter;
  while(RxFifo_Get(&letter) == FIFOFAIL){};
  return(letter);
}
```

e. The following code outputs one byte to the UART port.

```
// output ASCII character to UART
// spin if TxFifo is full
void UART_OutChar(char data){
  while(TxFifo_Put(data) == FIFOFAIL){};
  UART0_IM_R &= ~UART_IM_TXIM;          // disable TX FIFO interrupt
  copySoftwareToHardware();
```

```
  UART0_IM_R |= UART_IM_TXIM;          // enable TX FIFO interrupt
}
```

f. The system defines the ISR vector in the Startup.s file. Each exception has a 32-bit vector that points to the memory location where the ISR that handles the exception is located. Vectors are stored in ROM at the beginning of memory. All interrupts are enabled by setting the CPSIE bit.

g. The TM4C123 has large 16 byte hardware FIFOs like the PC.

2.

a. writedata uses Busy-Wait Synchronization.

```
void static writedata(uint8_t c) {
  while((SSI0_SR_R&SSI_SR_TNF)==0){};   // wait until transmit FIFO not full
  DC = DC_DATA;
  SSI0_DR_R = c;                        // data out
}
```

b. ST7735_DrawChar has five parameters:

x - horizontal of the top left corner of the character
y - vertical position of the top left corner of the character
c - character to be printed
textColor - 16-bit color of character
bgColor - 16-bit color of background
size - number of pixels per character pixel

To use this function, you pass the x,y position where you want the top left of the character to begin, the character to print, the color you would like to set the character, the color you would like to set as the background behind the character, and the size. You must keep in mind that x,y position is based on columns and the size parameter prints each pixel in the designated size.

c. Below are the pin assignments for two different versions of the ST7735 we are using.

/* Sain Smart Connections (ST7735)

Name	Pin	Board
VCC	1	+3.3V

GND	2	GND
SCL	3	PA2 (SSI0Clk)
SDA	4	PA5 (SSI0Tx)
RS/DC	5	PA6 (GPIO)
RES	6	PA7 (GPIO)
CS	7	PA3 (SSI0Fss)

Pins below are for the SD-Card:

MSO	8	nc
SCLK	9	nc
MOSI	10	nc
CS	11	nc

http://www.sainsmart.com/zen/documents/20-011-920/Manual.pdf

*/

/* Ada Fruit Connections (ST7735)

Name	Pin	Board
Backlight	10	+3.3V
MISO	9	nc
SCK	8	PA2 (SSI0Clk)
MOSI	7	PA5 (SSI0Tx)
TFT_CS	6	PA3 (SSI0Fss)
CARD_CS	5	nc
Data/Command	4	PA6 (GPIO)
RESET	3	PA7 (GPIO)
VCC	2	+3.3V
Gnd	1	GND

*/

d. PA2, PA5, and PA3 are only used for GPIO and the SSI function. One can use PA6 and PA7 as I2C and/or PWM functionality.

3.

a. The following code defines the period of the SysTick interrupt. PLL_Init() will define user specified clock speed, based on that, you will call SysTick_Init and pass it

a unsigned 32 bit value which is labeled as period which will then be used to set the reload value:

```
NVIC_ST_RELOAD_R = period-1;          // reload value
```

b. The clock is set by calling the PLL_Init() function. Specifically, the system clock is determined by the SYSDIV2 filed in the SYSCTL_RCC2_R register. Based on which number you select SYSDIV2 to be, the developer can change the clock cycle.

c. The SysTick interrupt is triggered when the CURRENT value counts down from 1 to 0, which sets the COUNT flag. The interrupt does not need to be acknowledged because it automatically reloads the CURRENT value and continues decrementing.

4.

a. The first assembly instruction is PUSH and the final instruction is BX LR.

b. It pushes eight registers on the stack: R0-R3, R12, LR, PC, PSR. R0 is on top and more registers are pushed if floating point is enabled.

c. When the ISR is complete, the LR contains a special value that that indicates a return from interrupt. Because the LR contains the special value, the BX LR instruction pulls the eight registers from the stack and returns control to the main program.