[**You may use one** 8.5×11 **inch reference sheet.**] Show all of your work clearly in the space provided. Be sure to **read each problem carefully**. Note that the exam is double sided.

1. #include

(a) (5 points) Describe what the #include directive does.

(b) (5 points) Explain the difference between #include "something.h" and #include <something.h>.

2. (5 points) Define what a stop bit is and explain the purpose for the stop bit(s) in the USART subsystem.

3. (5 points) Describe in words/pseudocode how to receive data using the USART subsystem in polling mode.

4. (10 points) Sketch the hardware configuration of the keypad and explain how why the keypad must be "scanned" in order to determine which key(s) is/are pressed.

5. (10 points) Write the C code necessary to set the two least significant bits and clear the two most significant bits in PORTB. The other bits should remain unchanged. You may not make any assumptions about the current configuration of the port.

6. (15 points) Implement the following method. You may assume that lcd_print_char(char x) has already been implemented. Note that the argument passed is a pointer instead of an array of characters (as it was in lab 3).

```
/**
  * Display a sequence of characters on the LCD panel beginning at
  * the cursor's current position.
  *
  * @param string A pointer to a null-terminated sequence of characters.
  */
void lcd_print_string(char* ptr) {



}
```

7. (5 points) Identify which ports (if any) the keypad should not be connected to and explain why?

8. (15 points) Implement the following method (assume that a variable, lcd_port exists and contains a lowercase character corresponding to the port to be written to):

```
/**
  * Sends data to port for the to which the LCD is attached.
  * If lcd_port contains a corrupt value, the default port (PORTC) will be used.
  *
  * @param value The value to be written to the port.
  */
void write_lcd_port(uint8_t data) {

}
```

9. (25 points) Write an entire C program that configures the LEDs to be placed on PORTB. The program should begin by turning all but one LED off. Whenever an external interrupt INT0 (vector name: INT0_vect) is triggered, the on LED should shift to the next most significant bit position. If the on LED is at bit 7, the ISR should force the on LED to bit 0, i.e., the on LED should wrap around. Whenever an external interrupt INT1 (vector name: INT1_vect) is triggered, the on LED should shift to the next least significant bit position. Similarly, if the on LED is at bit 0, the ISR should force the on LED to bit 7.
Your answer should include everything necessary in the .c file in order to generate the .hex file that could be downloaded to the ATmega32.

9. cont. . .