[**You may use one side of an** 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. (10 points) Describe the purpose of the interrupt jump table.

2. (10 points) The general purpose register R16 is treated as a *saved* register by the C compiler. Describe the steps (if any) that must be taken when using R16 in an assembly function that is called from C.

3. (10 points) State the type qualifier often needed with global variables modified by an ISR. Explain what the qualifer does.

4. (15 points) Show, in C, how to set the jump vector for the external interrupt request 0 to an interrupt service routine that sends the value of the timer/counter 0 subsystem's counter value to PORTB. You may assume that the port is already configured for output.

5. (15 points) Write the following C function:

```
/**
  * Swaps the values stored in the memory locations identified by
  * the pointers passed to the function.  If either pointer is a
  * null pointer, then no action is taken.
  * @param ptr1 Location of first uint8_t
  * @param ptr2 Location of second uint8_t
  */
void swap(uint8_t *ptr1, uint8_t *ptr2)
{









}
```

6. Consider the assembly function below:

```
exam:
     in  r1, SREG
     and r1, r24
     out PORTB, r1
     ret
```

(a) (5 points) This function could be called from C like this:

```
exam(0xff);
```

Explain what could go wrong if this function were called from C and suggest a change that would eliminate the potential problem.

(b) (10 points) Rewrite the assembly function as a C function.

7. (10 points) Implement the following function in C so that it can be called from assembly.

```
/* triplets -- Compares three unsigned 8-bit integers.  If all three
 *  values are the same, the function returns 1; otherwise 0 is returned.
 */
uint8_t triplets(uint8_t x, uint8_t y, uint8_t z);
```

8. (15 points) Implement the following function in assembly.

```
/* exclumultiore -- returns the bitwise exclusive or of all three arguments.
 */
uint16_t exclumultiore(uint16_t x, uint16_t y, uint16_t z);
```

The assembly instruction for exclusive or is:

```
EOR Rd, Rr
```