Name: Lab Section:

CprE 288 Fall 2012 – Homework 4
Due Thu. Sept. 20 in the class

Notes:

- **Start early on homework**
- Homework answers must be typed using a word editor. Hand in a hard copy in the class.
- Late homework is accepted within three days from the due date. E-mail the word file to your instructor. *Late penalty is 10% per day (counting from end of class on the due date).*

Question 1: Pointers (20 pts)

Complete the table below (i.e. fill in the memory map) to show the state of memory after the following C fragment has been executed. Assume the ATmega128 platform, and all the variables are in the stack (and thus the storage is allocated from high address to low address).

```
typedef struct pixel
{
  unsigned char red;
  unsigned char green;
  unsigned char blue;
  unsigned char psize;
} pixel_t;

char num_array[3] = {5, 10, 15};
unsigned char* ptr_array;
unsigned char x;
unsigned char y;
pixel_t my_pixel = {25, 15, 55, 40};
pixel_t  *pixel_ptr = &my_pixel;

ptr_array = num_array;
x = *ptr_array;
ptr_array++;
y = *ptr_array;
ptr_array = &num_array[2];
pixel_ptr++;
pixel_ptr++;
(*pixel_ptr).green = 5;
```

Address	Variable Name	Value
1000	num_array[2]	15
999	num_array[1]	5
998	num_array[0]	5
997	ptr_array	
996		1000
995	x	5
994	y	10
993	my_pixel.psize	40
992	my_pixel.blue	55
991	my_pixel.green	15
990	my_pixel.red	25
989	pixel_ptr	
988		998

Name: Lab Section:

Question 2: Struct and Union (10 pts)

What is the size of the following data structures in bytes, assuming we're using the Atmel compiler?

a) 2 pts.

```
struct point3D {
      long x;                 4
      unsigned long y;        4
      char z;                 1
};
```

Size of struct is sum of components: 4 + 4 + 1 = 9 bytes

b) 2 pts.

```
union val {
      char cval;        1
      short sval;       2
      int ival;         2
      float fval;       4
      double dval;      4
};
```

Size of union is the size of the largest component: 4 bytes

c) 3 pts.

```
struct compound {
      char *mystring;    2
      union {
            char c;
            int i;
            float f;     4
      };
};
```

Size of struct is sum of components: 2 + 4 = 6 bytes

d) 3 pts.

```
struct more_compound{
      char *myname;      2
      int  *myage;       2
      long mypay;        4
      long *myspeed;     2
      union {
            char *c;
            int  *i;
            long *l;     2
      };
      char red;          1
      char green;        1
      char blue;         1
      union {
            char my_c;
            int  my_i;
            long my_l;   4
      };
}
```

Size of struct is sum of component sizes:
2 + 2 + 4 + 2 + 2 + 1 + 1 + 1 + 4 = 19 bytes

Name: Lab Section:

Question 3: Bitwise operations (10 pts)

a.) Let **x** be an 8-bit variable (of type char). The following exercises test your knowledge of bit masking. For the following, **write a single line of C code that represents the given description**. The first has been completed for you. (5 pts)

i) Description: if bit 0 of x is set
if (x & 0x01)

ii) Description: while bit 5 and 4 of x is clear
while(!(x & 0x30))

iii) Description: clear the upper 4 bits of x while preserving the lower 4 bits
x = x & 0x0F;

iv) Description: toggles bits 7, 6, 3, and 2 of x while persevering the rest
x = x ^ 0b11001100 in hex x = x ^ 0xCC

v) Description: clear bit 3 and set bit 7 while preserving the other bits
x = (x & 0xF7) | 0x80;

b) Using bitwise operators, shift operators, and a single for loop, complete the following C function to pack the one-value bits in a variable **x,** of type unsigned short, into bit positions 0 to "number of ones"-1. (5 pts)

Example:

```
unsigned short x = 0b0001000010101010;  // has 5 ones
unsigned short y = 0;
y = pack_ones(x);
```

// After calling pack_ones in this case:
// y = 0b0000000000011111, the 5 ones are packed from bit positing 0 to bit position 4 (i.e. 5 – 1);

```
unsigned short pack_ones(unsigned short x) {
      int i = 0;
      int packed_x = 0;

      for(i = 0; i < 16; i++)
      {
            if( x & 0x01)  // check if bit 0 is a one
            {
                  packed_x = packed_x << 1;
                  packed_x = packed_x | 0x01; //add a one to packed_x
            }
            x = x >> 1; //shift so the next bit position can be checked
      }
return packed_x;
}
```

Name: Lab Section:

Question 4: IO Ports (20 pts)

We wish to drive a series of LEDs using the ATmega128 ports in order to create a binary clock.

Specifications
The LED clock consists of 16 LEDs aligned in three rows. Binary LED numbers are represented in little endian form (i.e. the least significant bit is the rightmost one). There is no LED to distinguish between AM or PM.

- The top row of 4 LEDs gives the binary representation of the hour of the day (1 through 12).
- The middle row of 6 LEDs gives the binary representation of the current minute (0 through 59).
- The bottom row of 6 LEDs gives the binary representation of the current second (0 through 59).

The LEDs will be driven using ports A and B of the ATmega128. The following lists the mapping between bits on PORTA and PORTB and the LEDs on the clock. A high signal will light the LEDs.

PORTA = $h_3h_2h_1h_0m_3m_2m_1m_0$
PORTB = $s_5s_4s_3s_2s_1s_0m_5m_4$

Thus, to display a time of 1:35:02 pm:
PORTA = 0b00010011;
PORTB = 0b00001010;

There is an additional LED connected to bit6 of port D. We will use this LED as an alarm. It will turn on when the clock equals an alarm setting

Given these specifications of the LED binary clock, **complete the following three functions**.

```
// Initialize the ports
void initClock() {
  // Set ports to be input or output using Data Direction Registers
        DDRA = 0xFF; // All bits of Port A set for output
        DDRB = 0xFF; // All bits of Port B set for output
        DDRD = DDRD & 0xBF; // Set bit 6 of Port D for input, preserved configuration of other bits
}

// Drives the ports to make the LEDs turn on or off
void setTime(unsigned int hours, unsigned  int minutes, unsigned int seconds) {

        PORTA = (hours << 4) |  (minutes & 0x000F);
        PORTB = (seconds << 2) | (minutes >> 4);

}
```

Name: Lab Section:

```
// Drive the alarm LED if the current time is alarm_time when this function is called.
void triggerAlarm(unsigned int alarm_time){
     // Make sure bits of alarm_time are packed consistently with how they are packed for the Ports.
     // In this case I have assumed:
     // The upper 8-bits of alarm_time hold PORTA bits (i.e. h3h2h1h0m3m2m1m0) and
     //  The lower 8-bits of alarm_time hold PORTB bits  (i.e.  s5s4s3s2s1s0m5m4  )
     //  With the above assumption just align bits and check for equality.  It does not matter how you
    // pack the bits of alarm_time as long as when you check you are consistent with how PortA and
    // PortB are packed.

        unsigned int check_portA = 0;
        unsigned int check_portB = 0;
        check_portA = PINA;
        check_portB = PINB;

        if(  ((alarm_time >> 8) == check_portA)  &&  ((alarm_time & 0x00FF) == check_portB)  )
        {
                PORTD = PORTD | 0x40;  // Set bit 6 high, preserved the other bits of PortD
        }

}
```