CprE 288 Spring 2014 – Homework 5
Due Fri. Feb. 21

Notes:
• Homework answers must be typed using a word editor.
• Homework is individual work. Adhere to the University’s policy relating to the integrity of scholarship. See http://catalog.iastate.edu/academiclife/#regulationstext, “Academic Dishonesty”.
• Late homework is accepted within three days from the due date. Late penalty is 10% per day (for each 24 hours following your class).

Note: Unless otherwise specified, all problems assume the ATMega128 is being used

Question 1: C Operator Precedence (15 pts)

Does each of the code pieces serve for its stated purpose? You must explain to get the full credit. Assume the following variable declaration:

```c
unsigned int a, b, c, d, mask;
unsigned int *p = &a;
```

Hint: Check the operator precedence chart, slide 49 of week 5.

a. [3] Check if a is equal to b, or b is equal to c and c is equal to d.

```c
if (a == b || b == c && c == d)
    ... // code
```

Yes, because == higher than &&, and && is higher than ||

b. [3] Check if the upper half of a is equal to the upper half of b.

```c
mask = 0xFF00;
if (a&mask == b&mask)
    ... // code
```

No, because == will run first. == has higher precedence than &.


```c
b = *p++;
```

Yes, because p++ will run first and * dereference what p points at and assign it into b.
d. [3] Add the lower half of a and the lower half of b, store the result to c. The result should appear in the lower half of c, and the upper half of c should be cleared to zero.

\[ c = (a \ll 8 + b \ll 8) \gg 8; \]

No, + has higher precedence than <<.

e. [3] Swap the upper half and the lower half of b.

\[ b = b \ll 8 \mid b \gg 8; \]

Yes, \( \ll \) and \( \gg \) has higher precedence than \( \mid \).

**Question 2: C Type Conversion and Casting (15 pts)**

What is the outcome of each of the following assignment statements or if-condition? For an assignment statement, state the value of the variable being assigned after the execution of the statement. You must explain to get the full credit. Assume the following variable declaration:

```c
signed char byte = -50;
short ADC_input = 500;
unsigned int n;
```

*Hint: If you are not sure about an outcome, you can write a program to find out (using AVR studio).*

a. [3] byte = ADC_input;

byte = 0xF4 because byte only have one byte. If assign 2 bytes of 0x01F4, then the byte will be 0xF4.

b. [3] n = (unsigned int) byte;

n = 0x00ce because force cast byte to int will give 0x00ce.

c. [3] n = (int) byte;

n = 0xffce, because byte is an negative number , when it force cast into integer, it will add 1 in front of the original bits.
d. \[3\] \( n = \text{ADC}_\text{input} + (\text{signed char}) \text{ byte}; \]

\( n = 0x01c2, \text{ because } 500 \text{ minus } 50 \text{ is equal to } 450 = 0x01c2 \)

e. \[3\] \( n = \text{ADC}_\text{input} + (\text{unsigned char}) \text{ byte}; \)

\( n = 0x02c2, \text{ because } -50 \text{ is } 0xce \text{ when force into unsigned char. And } 0x01f4 + 0xce \text{ is equal to } 0x02c2 \)

**Question 3: C Standard Library (10 pts)**

Assume in a programming assignment you need a function to **test whether a character appears in a C string**; for example, yes for ‘C’ being a character in “This is CprE288”. Find out such a function in C Standard Library. Write a short program to verify that the function you found works. Test it in a programming environment, e.g. Linux, Microsoft Visual Studio, or Dev-C++ (use only C code). Cut and paste your program code and test output here.

**Code:**

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char * str = "This is CprE288";
    char c = 'C';
    char b = 'B';
    if(strchr(str,c)!="")
        printf("%c is in %s\n",c,str);
    else
        printf("%c is not in %s\n",c,str);
    if(strchr(str,b)!="")
        printf("%c is in %s\n",b,str);
    else
        printf("%c is not in %s\n",b,str);
}
```

![Image of the code and output](image-url)
Question 4: Memory-Mapped I/O (10 pts)

Consider developing memory-mapped interface for a mouse along the lines of the keyboard example presented in the class.

a. Specify the data and control register(s) in this interface. Assume that the mouse transmits \((\Delta X, \Delta Y)\), (the distance the mouse has traveled since last click in byte sized chunks) and left or right click occurrences.

Data:

```c
char* MSDR;
```

Control register:

```c
struct MSCR{
    unsigned int x;
    unsigned int y;
    unsigned char traveled;
    unsigned int left;
    unsigned int right;
    unsigned int READY;
}
```

MSCR:
b. Assume that the I/O address space starts at 0xFF00. Assign addresses to all the data and control registers in the memory-map interface of the mouse.

```
MSDR = 0xFF00;
MSCR = 0xFF01;
```

c. Show a polled version of a function `readDeltaX()` that reads \( \Delta X \) into a program variable `deltaX`. Only fresh data should be read as in the keyboard example in the class.

```
readDeltaX()
{
    Int oldx = MSCR.x;
    while(!(MSCR.READY));
    deltaX = MSCR.x - oldx;
    return;
}
```

d. If we want to allow the mouse to have the capability to interrupt the processor, what else would have to be added to its memory map interface? Show an ISR equivalent of `readDeltaX()` that is bound to the interrupt vector `CLICK_DELTA_X_VECT`.

```
ISR (CLICK_DELTA_X_VECT)
{
    int oldx = MSCR.x;
    if(MSCR.READY){
        deltaX = MSCR.x - oldx;
    }
}
```

**Question 5: Interrupt Service Routine (10 pts)**

Explain why a variable shared by an ISR and the foreground program execution must be declared as “volatile”. In particular, discuss what will happen if “volatile” is NOT used on the shared variable and compiler optimization is enabled. The following is an example:

```
volatile static int clock_flag = 0;

ISR (TIMER1_COMPA_vect)
{
    clock_flag = 1;
}
```

```
int main()
```
{  
  ... // other code

  while (1)
  {
    if (clock_flag == 1)
    {
      ... // advance the clock, print clock on LCD
    }
  }
}

Volatile is intended to prevent the compiler from applying certain optimizations that make the program running in the way you don’t mean to be. When volatile is not used, in the while loop, the clock_flag will be always in 0 state, even if the interruption is applied. So, without it, the program will not print clock on LCD forever.