ICS'12 Spring Homework 1

I Number Conversion

i. Convert the following numbers with the indicated bases to decimal.

$(E1B7)_{16}$ $(110110011010)_2$ $(753)_8$

ii. Convert the following numbers with the indicated bases to hexadecimal.

$(65535)_{10}$ $(100101111010)_2$ $(777)_8$

iii. Convert the following numbers with the indicated bases to binary.

$(64030)_{10}$ $(ABCD)_{16}$ $(567)_8$

II Operations

i. Given A and B with hexadecimal expression 0xAB and 0xD2 respectively. Calculate the values of the following expressions:

a) A & B

b) A && B

c) A | B

d) A || B

e) A ^ B

f) ~A | ~B

ii. Fill in the table below with the results of shift operation given

X	X<<4	X>>3(Logical)	X>>3(Arithmetic)
0xB2			
0x0F			
0xDC			
0xA9			

iii. Design C expressions using bitwise (& | ~ ^) and logical (&& || !) operators, which return 1 under the specific conditions described below, otherwise return 0.

A. all bits of x are 1.

B. all bits of x are 0.

C. bits of x's least significant byte are 1.

D. bits of x's least significant byte are 0.

iv. Complete the following C function (as simple as possible) to set a specific bit of a 32-bit signed integer to the given value. ONLY bitwise (&, |, ~, ^) and shift (<<, >>) operators are allowed. Please also describe your expression briefly.

```
/*   Example:
     * set_bit(0x55766ab8, 15, 1)==0x5576eab8
     * set_bit(0x55766ab8, 11, 0)==0x557662b8
*/
int set_bit(int number, int position, bool bit)
{
    return _____;
}
```

v. Design a C expression, which generates a word (32-bit) consisting of the least significant byte of x and the remaining bytes of y.

For example, x = 0x89ABCDEF and y = 0x76543210, it will generate 0x765432EF.

III Byte Ordering

i. Consider the following function:

```
typedef unsigned char *byte_pointer;

void show_bytes(byte_pointer start, int len)
{
```

```
    int i;
    for (i = 0; i < len; i++)
        printf("%.2x", start[i]);
    printf("\n");
}
```

What is the output of the following call to *show_bytes* on big-endian and little-endian machines respectively?

```
int val = 0x78563412
byte_pointer valp = (byte_pointer) &val;
show_bytes(valp, 1);    /* A. */
show_bytes(valp, 2);    /* B. */
show_bytes(valp, 3);    /* C. */
show_bytes(valp, 4);    /* D. */
```

A. little-endian:_____ big-endian:_____
B. little-endian:_____ big-endian:_____
C. little-endian:_____ big-endian:_____
D. little-endian:_____ big-endian:_____

ii. Implement a function *is_little_endian()*, which returns 1 if it is running on little-endian machine and 0 if it is running on big-endian machine. This function should be able to run on any machine regardless of the difference of word size.