Homework 2

I. Byte Order

1. Consider the following function [`sizeof(int) == 4`]

```
typedef unsigned char byte;

void show_bytes(byte *start, int len) {
    int i;
    for(i = 0; I < len; i++)
        printf("%.2x", start[i]);
}

unsigned int val = 0x87654321;
byte *valp = (byte *) &val;
```

(1) What is the value of valp[4] (if consider it as an array of 4 elements of "`byte`")?
Little-endian: {valp[0], valp[1], valp[2], valp[3]} = {0x21, 0x43, 0x65, 0x87}
Big-endian: {valp[0], valp[1], valp[2], valp[3]} = {0x87, 0x65, 0x43, 0x21}

(2) What is the output of the following call to show_bytes on big-endian and little-endian machines respectively? (You can have a try on your machine!)

	Little-endian	Big-endian
show_bytes(valp, 1)	21	87
show_bytes(valp, 2)	2143	8765
show_bytes(valp, 4)	21436587	87654321

2. Write a procedure `is_big_endian` that will return 1 when compiled and run on a big-endian machine, and will return 0 when compiled and run on a little-endian machine. This program should run on any machine, regardless of its word

```
int is_big_endian(void) {
    // fill in C codes …
    // not the only answer
    int x = 1;
    return 1 – (int)(*(char *)&x);
}
```

3. Write a procedure `to_big_endian` that will transform an 32 bit unsigned integer number into big endian format and return the pointer to the big endian integer. This program should run on both little endian and big endian machine. (Hint: you can use `is_big_endian` function written before)

```
byte *to_big_endian(unsigned int num) {
    // fill in C codes …
    // not the only answer
    /* directly return pointer to local variable can also be
```

```
considered right at this time */
    byte *ret = (byte *)malloc(sizeof(int));
    if (is_big_endian()) {
        memcpy(ret, &num, sizeof(int));
    } else {
        byte *bytes = (byte *)&num;
        ret[0] = bytes[3];
        ret[1] = bytes[2];
        ret[2] = bytes[1];
        ret[3] = bytes[0];
    }
    return ret;
}
```

J. Encodings

4. Integer representation using 1's and 2's complement. Assume we have an integer type of 8 bits, fill in the following table.

Value(Decimal)	2's complement	1s' complement	Sign-Magnitude
19	0001 0011	0001 0011	0001 0011
-51	1100 1101	1100 1100	1011 0011
-110	1001 0010	1001 0001	1110 1110
-84	1010 1100	1010 1011	1101 0100
-42	1101 0110	1101 0101	1010 1010
x	~x+1	~x	x ^ (1 << 8)