Assignment 1: Bit Operation

Modify the provided skeleton code and implement the following five functions as shown below.

- You should use bit operations to solve the problems.
- Do not modify function declaration.
- You can not use any C/C++ library functions other than malloc and free. Do not include any other header files.
- All the implementation should be executed without any error if there is no assumption, e.g., your implementation should check all corner cases that may cause segmentation faults.
- Assume the “pointer” type is defined as “unsigned char *”. (Provided with the skeleton)

```
void reverse_bit(pointer a, int len); (2pt)
void inverse_bit(pointer a, int len); (2pt)
void split_bit(pointer a, pointer out1, pointer out2, int len); (2pt)
unsigned short partial_mul(unsigned short a, unsigned short b); (2pt)
void get_date(unsigned int date, int* pYear, int* pMonth, int* pDay);  (2pt)
```

Due

Submit your implementation before Apr 2, Friday, 11:59:59pm, to LMS. We DO NOT allow late submission.

Submission

Submit the cpp file that has your implementation. You have to implement all the functions in a single cpp file. Before the submission, rename the cpp file with this format: “hw1_YOURSTUDENTID.cpp”. For example, “hw1_200012345.cpp”

Function Specification

1. void reverse_bit(pointer a, int len);
    a. Reverse the bit order for the data stored in a
    b. e.g., for two-byte data, a = $1111000010111101_{(2)}$, the function manipulates it by $1011110100001111_{(2)}$.
    c. Parameters:
        i. a: data to be reversed (byte array)
        ii. size: the size of a in bytes

2. void inverse_bit(pointer a, int len);
    a. Inverse each bit state for the data stored in a
    b. e.g., for e.g., for two-byte data, a = $1111000010111101_{(2)}$, the function manipulates it by $0000111101000010_{(2)}$.
    c. Parameters:
        i. a: data to be inversed (byte array)
        ii. size: the size of a in bytes

3. void split_bit(pointer a, pointer out1, pointer out2, int len);
    a. For the data stored in a, take all odd bits and store it into out1 / take all even bits and store it into out2
    b. e.g., for e.g., for two-byte data, a = $1111000010111101_{(2)}$, the function stores $11001110_{(2)}$ and $11000111_{(2)}$ to out1 and out2, respectively.
    c. Assume the length of a, i.e., len, is multiplies of 2. Also assume that the byte lengths in out1 and out2 are len/2.
    d. Parameters:
        i. a: data to be split (byte array)
        ii. out1 / out2: odd/even bits as the outputs
        iii. len: the size of a in bytes

4. unsigned short partial_mul(unsigned short a, unsigned short b);
    a. Return the multiplication of 0th~5th bits of a and 8th~14th bits of b
    b. e.g.,

a =

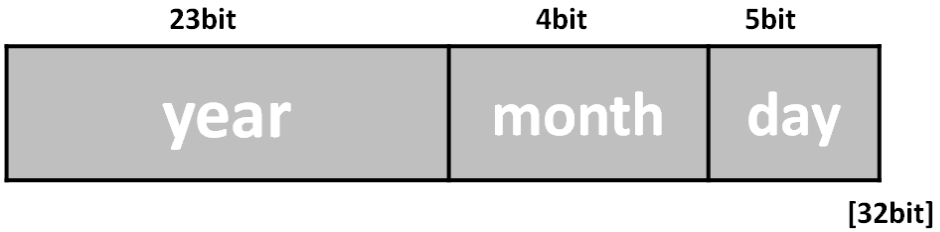
Index	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Num	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0	1

b =

Index	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Num	1	1	1	1	0	0	0	0	1	0	1	1	1	1	0	1

$111101_{(2)}$ X $1110000_{(2)}$=$1101010110000_{(2)}$=$6842_{(10)}$

5. void get_date(unsigned int date, int* pYear, int* pMonth, int* pDay);
    a. Assume the date is encoded with an unsigned integer, date, as follows:



    b. Decode it and store the results into pYear, pMonth and pDay.

Sample

Note: We will check with other example inputs for grading.

Sample Code

```
int main() {
    printf("Problem 1\n");
    unsigned int v1 = 0x1234CDEF;
    print_bit((pointer)&v1, sizeof(v1));
    reverse_bit((pointer)&v1, sizeof(v1));
    print_bit((pointer)&v1, sizeof(v1));


    printf("Problem 2\n");
    unsigned int v2 = 0x1234CDEF;
    print_bit((pointer)&v2, sizeof(v2));
    inverse_bit((pointer)&v2, sizeof(v2));
    print_bit((pointer)&v2, sizeof(v2));


    printf("Problem 3\n");
    unsigned int v3 = 0x1234CDEF;
    unsigned short out1 = 0, out2 = 0;
    print_bit((pointer)&v3, sizeof(v3));
    split_bit((pointer)&v3, (pointer)&out1, (pointer)&out2, sizeof(v3));
    print_bit((pointer)&out1, sizeof(out1));
    print_bit((pointer)&out2, sizeof(out2));


    printf("Problem 4\n");
    unsigned short v4 = 0xF0BD;
    print_bit((pointer)&v4, sizeof(v4));
    unsigned short v4_out = partial_mul(v4, v4);
    print_bit((pointer)&v4_out, sizeof(v4_out));


    printf("Problem 5\n");
    unsigned int date = 1034867;
    int year, month, day;
    get_date(date, &year, &month, &day);
```

```
    printf("%d -> %d/%d/%d\n", date, year, month, day);

    return 0;
}
```

Output

```
Problem 1
11101111 11001101 00110100 00010010
01001000 00101100 10110011 11110111
Problem 2
11101111 11001101 00110100 00010010
00010000 00110010 11001011 11101101
Problem 3
11101111 11001101 00110100 00010010
11111010 01000001
10111011 01100100
Problem 4
10111101 11110000
10110000 00011010
Problem 5
1034867 -> 2021/3/19
```