```c
/*
 * Filename: Assn1_Q3.c
 *
 * Description:
 *
 * Author: AL + Jai Malhi
 * Student number: 301457742
 */

#include <stdio.h>
#include <stdlib.h>

typedef unsigned char *byte_pointer;

// Question 3 a.
void show_bytes(byte_pointer start, size_t len){
    printf("show_bytes: \n");
    size_t i;
    for (i = 0; i < len; i++){
        printf("%p: %.2x\n", &start[i], start[i]);
        //printf(" %.2x", start[i]); //Original
    }
    //%p is used to print the memory address of the byte and %.2x
    //is used to print the content of the byte in hexadecimal format.
    //The & operator is used to take the address of the byte at
    //the current index of the loop.
    printf("\n");
    return;
}

// Question 3 b.
// Put your answer to Question 3 b) here as a comment
/*
for number 12345 I got: 00003039
    0x7ffc85558bac: 39
    0x7ffc85558bad: 30
    0x7ffc85558bae: 00
    0x7ffc85558baf: 00
the most significate bit is stored at 0x7ffc85558baf and least significate
bit is at 0x7ffc85558bac which indicated that our computer is little endian

for a negative number -12345 I got: ffffcfc7
    0x7ffc5468fd2c: c7
```

```
    0x7ffc5468fd2d: cf
    0x7ffc5468fd2e: ff
    0x7ffc5468fd2f: ff
again little endian since MSB is at 0x7ffc5468fd2f and LSB is at 0x7ffc5468fd2c
*/


// Question 3 c.
void show_bytes_2(byte_pointer start, size_t len){
    printf("show_bytes_2: \n");
    size_t i;
    for (i = 0; i < len; i++){
        printf("%p: %.2x\n", start+i, *(start+i));
        //printf(" %.2x", start[i]); //Original
    }
    printf("\n");
    return;
}

// Question 3 d.
void show_bits(int decimal){
    //array to store binary number
    int binaryNum[32];
    int temp = decimal;
    int i = 0;
    int j = 31;

    //count bit pattern via modulus
    for(i=0; i <=31; i++){
        //storing value in array
        binaryNum[i] = abs(temp % 2);
        temp = temp / 2;
    }
    //negative decimal
    if(decimal < 0){
        //find first 1 from left
        for(j = 0;j <= 31; j++){
            if(binaryNum[j] == 1){
                break;
            }
        }
        //add padding based on 1 or 0
        for(int k = j + 1; k <= 32; k++){
```

```
            if(binaryNum[k] == 1){
                binaryNum[k] = 0;
            }
            else if(binaryNum[k] == 0){
                binaryNum[k] = 1;
            }
        }
    }

    //printing out the resulting bit pattern
    for (j = 31;j >= 0; j--){
        printf("%d", binaryNum[j]);
    }
    printf("\n");
}


// Question 3 e.
int mask_LSbits(int n){ //TODO: make your own
    int mask;
    //if n <= 0, returning a mask of all 0s
    if(n <= 0){
        mask = 0;
    }
    //if n >= 32 returning a mask of all 1s
    else if (n >= 32){
        mask = - 1;
    }
    //left shift 1 by n, new value equals 2^n, then subtract 1 to get proper
value
    else {
        mask = (1 << n) - 1;
    }
    //* Testing
    //printf("%d\n", mask); //prints the mask in decimal form
    //printf("0x%x\n", mask); //prints the mask in hexadecimal form
    return mask;
}

void show_int(int x) {
    printf("============================");
    printf("\nival = %d\n", x);
    show_bytes((byte_pointer) &x, sizeof(int));
```

```
    show_bytes_2((byte_pointer) &x, sizeof(int));
    return;
}

void show_float(float x) {
    printf("============================\n");
    printf("fval = %f\n", x);
    show_bytes((byte_pointer) &x, sizeof(float));
    show_bytes_2((byte_pointer) &x, sizeof(float));
    return;
}

void show_pointer(void *x) {
    printf("============================\n");
    printf("pval = %p\n", x);
    show_bytes((byte_pointer) &x, sizeof(void *));
    show_bytes_2((byte_pointer) &x, sizeof(void *));
    return;
}
```