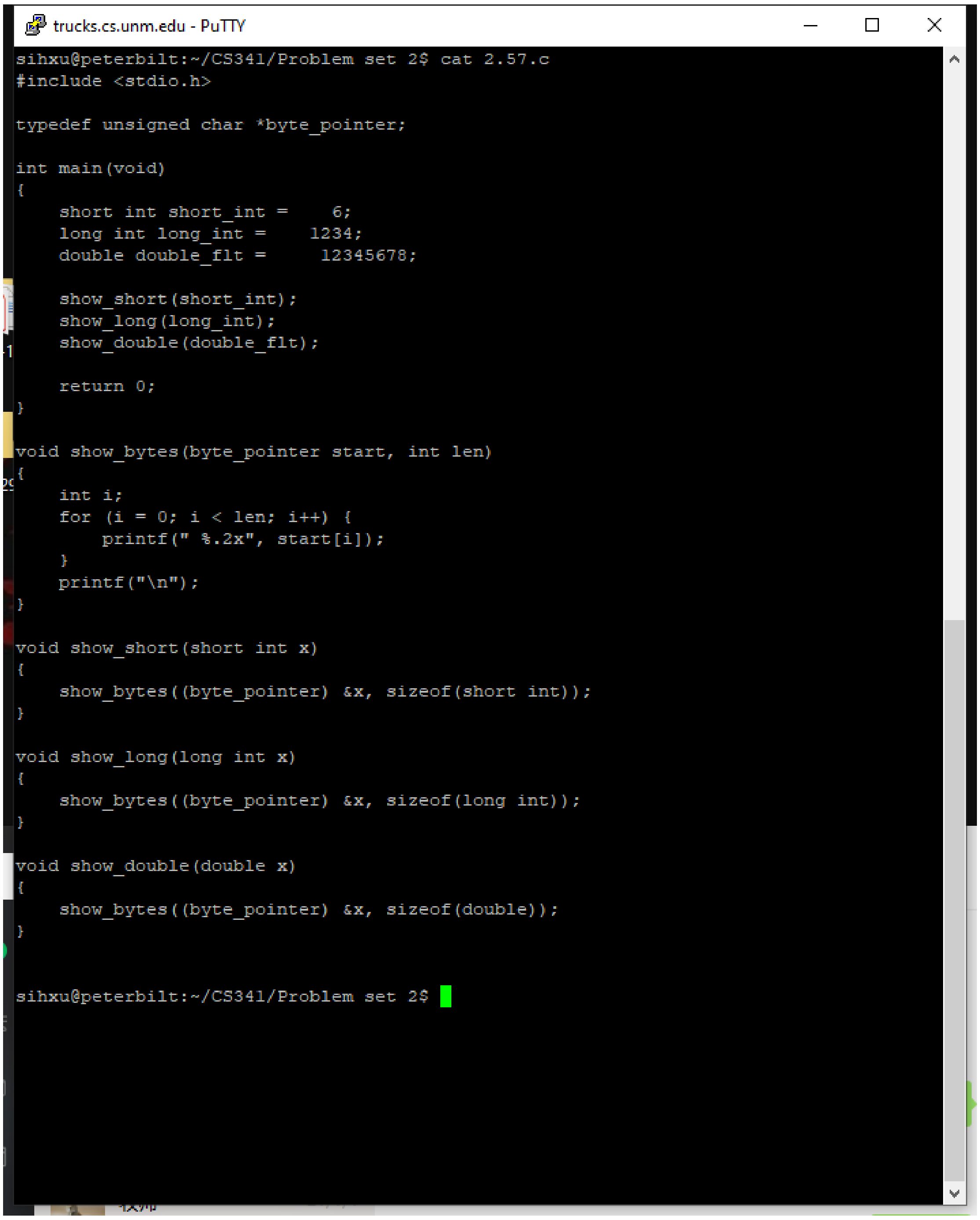
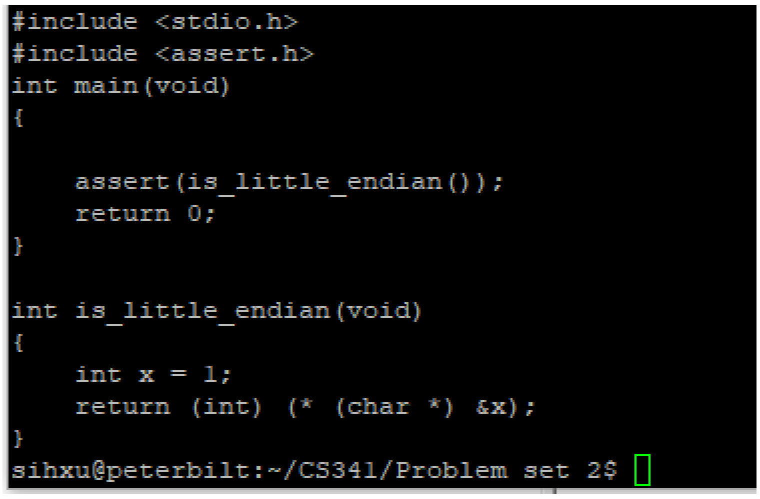
2.57



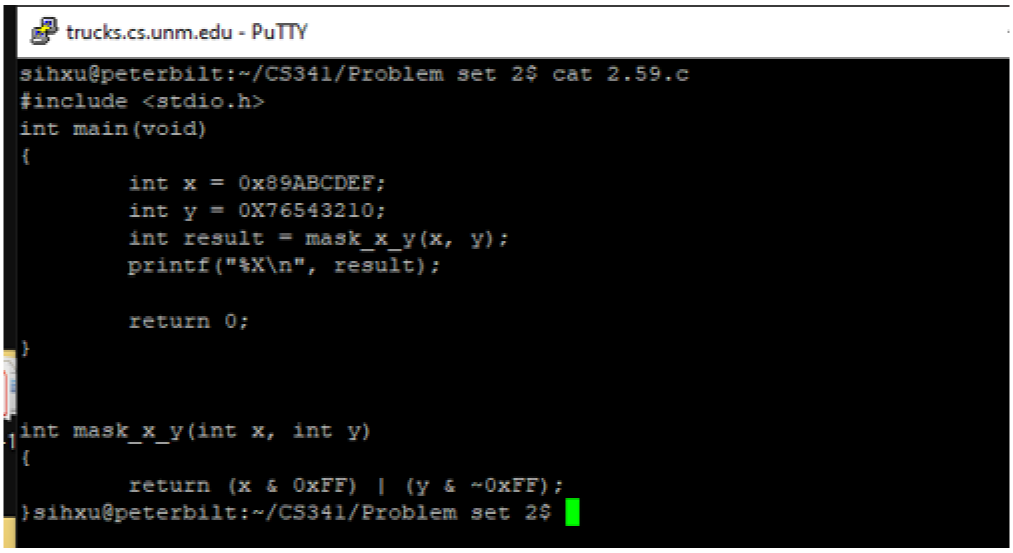
```
sihxu@peterbilt:~/CS341/Problem set 2$ cat 2.57.c
#include <stdio.h>

typedef unsigned char *byte_pointer;

int main(void)
{
    short int short_int =    6;
    long int long_int =    1234;
    double double_flt =     12345678;

    show_short(short_int);
    show_long(long_int);
    show_double(double_flt);

    return 0;
}

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

void show_short(short int x)
{
    show_bytes((byte_pointer) &x, sizeof(short int));
}

void show_long(long int x)
{
    show_bytes((byte_pointer) &x, sizeof(long int));
}

void show_double(double x)
{
    show_bytes((byte_pointer) &x, sizeof(double));
}


sihxu@peterbilt:~/CS341/Problem set 2$
```

2.58



```
#include <stdio.h>
#include <assert.h>
int main(void)
{

    assert(is_little_endian());
    return 0;
}

int is_little_endian(void)
{
    int x = 1;
    return (int) (* (char *) &x);
}
sihxu@peterbilt:~/CS341/Problem set 2$ 
```

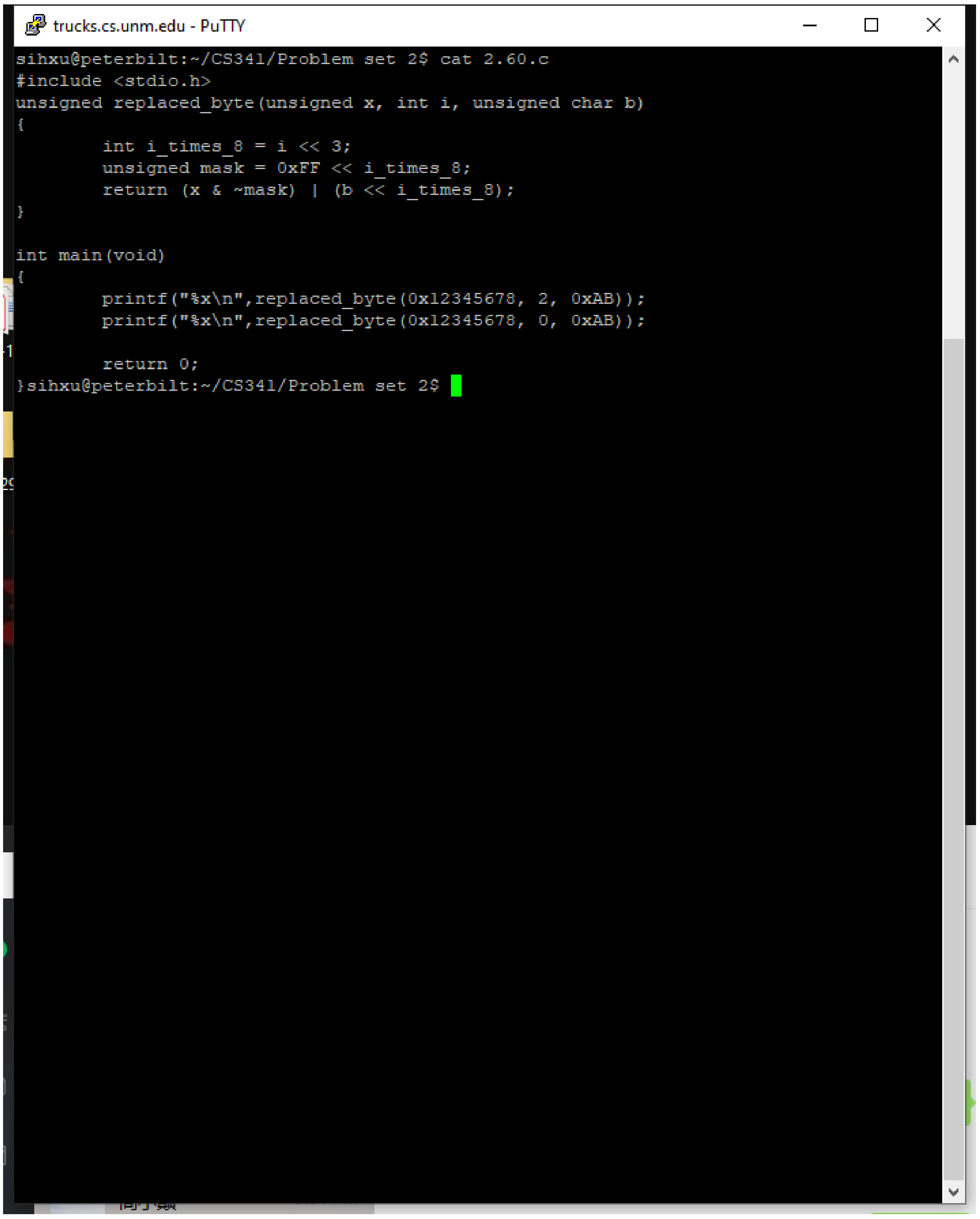
2.59



trucks.cs.unm.edu - PuTTY

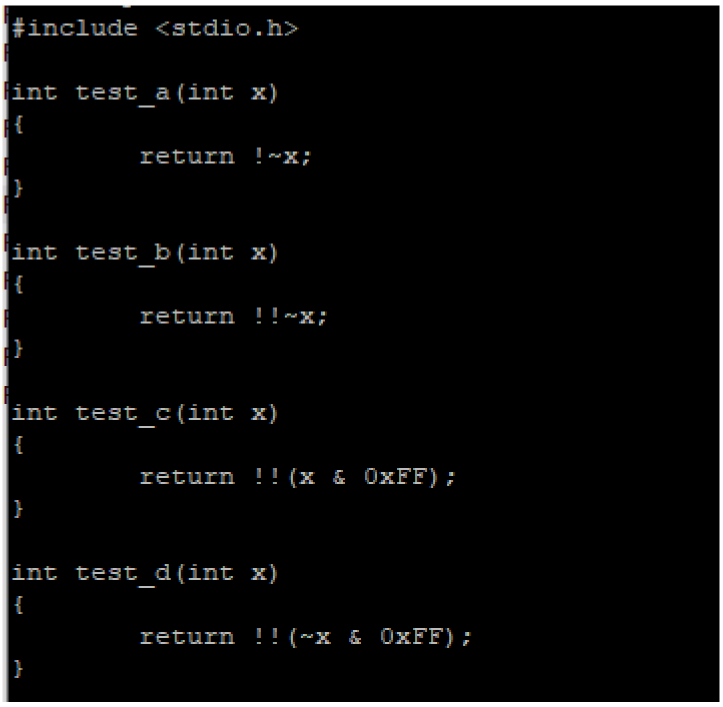
```
sihxu@peterbilt:~/CS341/Problem set 2$ cat 2.59.c
#include <stdio.h>
int main(void)
{
        int x = 0x89ABCDEF;
        int y = 0X76543210;
        int result = mask_x_y(x, y);
        printf("%X\n", result);

        return 0;
}


int mask_x_y(int x, int y)
{
        return (x & 0xFF) | (y & ~0xFF);
}sihxu@peterbilt:~/CS341/Problem set 2$ 
```

2.60



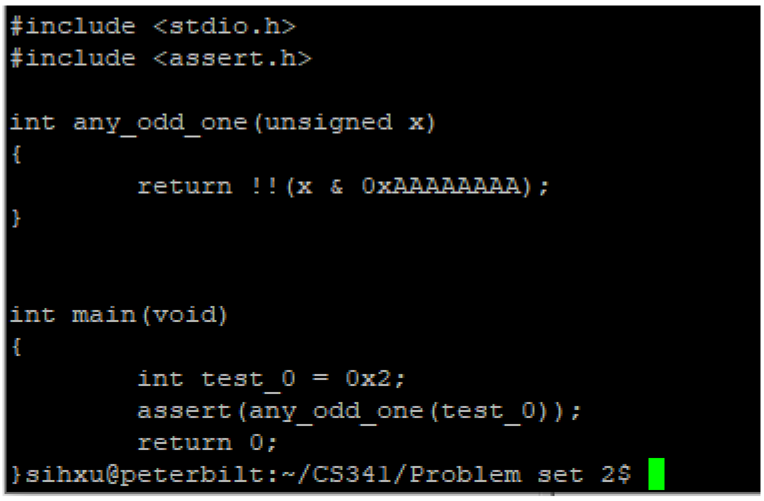
```
sihxu@peterbilt:~/CS341/Problem set 2$ cat 2.60.c
#include <stdio.h>
unsigned replaced_byte(unsigned x, int i, unsigned char b)
{
        int i_times_8 = i << 3;
        unsigned mask = 0xFF << i_times_8;
        return (x & ~mask) | (b << i_times_8);
}

int main(void)
{
        printf("%x\n",replaced_byte(0x12345678, 2, 0xAB));
        printf("%x\n",replaced_byte(0x12345678, 0, 0xAB));

        return 0;
}sihxu@peterbilt:~/CS341/Problem set 2$
```

2.61



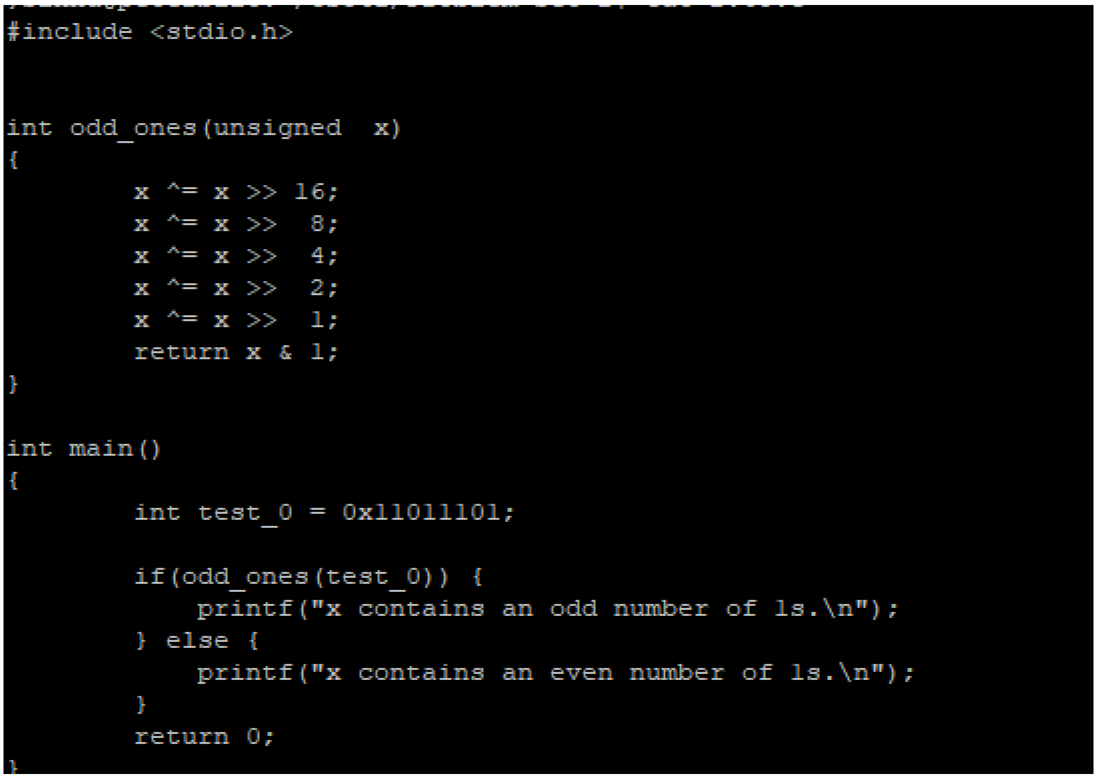
```
#include <stdio.h>

int test_a(int x)
{
        return !~x;
}

int test_b(int x)
{
        return !!~x;
}

int test_c(int x)
{
        return !!(x & 0xFF);
}

int test_d(int x)
{
        return !!(~x & 0xFF);
}
```

2.64



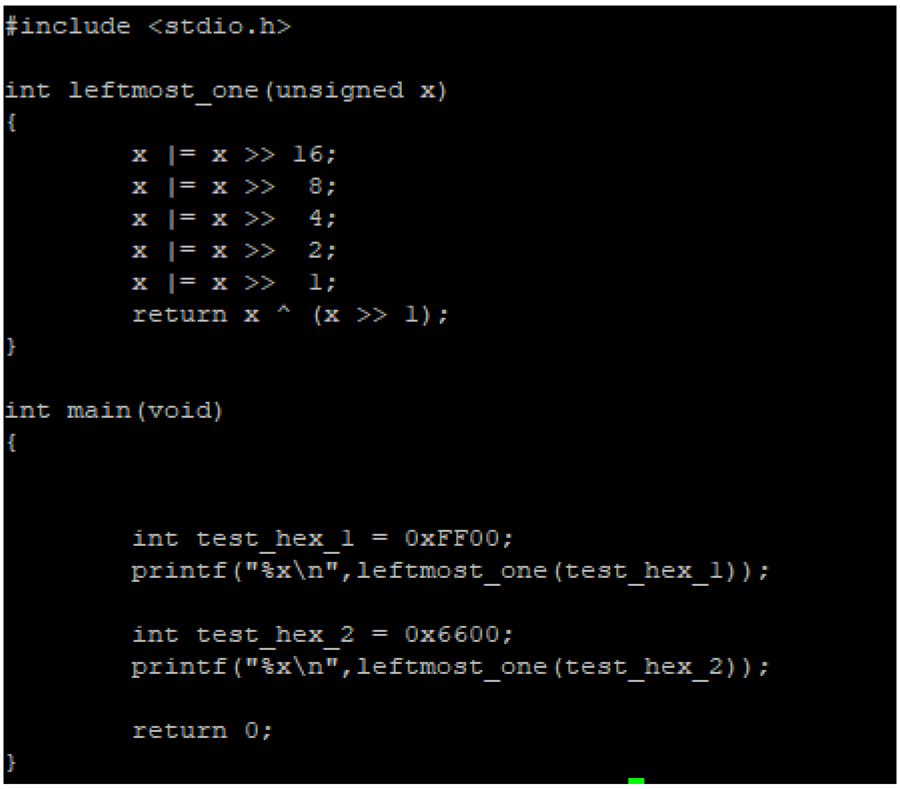
```
#include <stdio.h>
#include <assert.h>

int any_odd_one(unsigned x)
{
        return !!(x & 0xAAAAAAAA);
}


int main(void)
{
        int test_0 = 0x2;
        assert(any_odd_one(test_0));
        return 0;
}sihxu@peterbilt:~/CS341/Problem set 2$
```

2.65



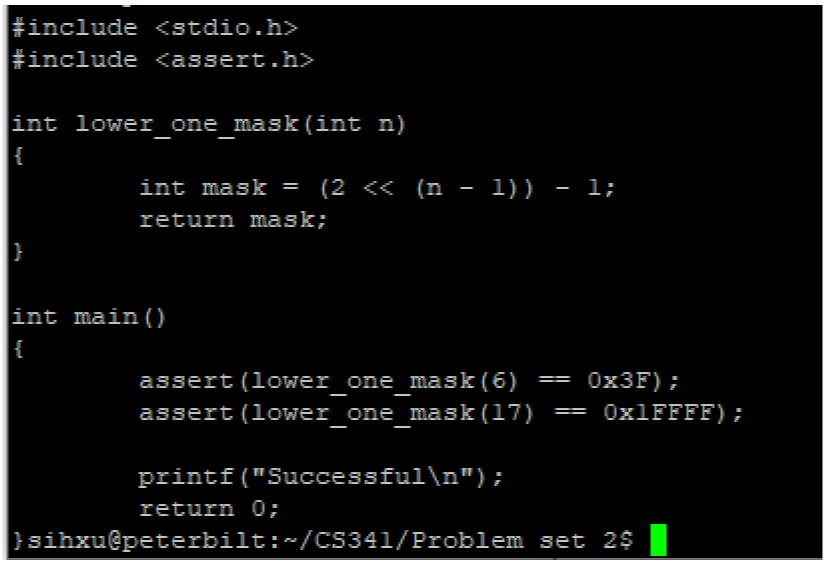
```c
#include <stdio.h>


int odd_ones(unsigned  x)
{
        x ^= x >> 16;
        x ^= x >>  8;
        x ^= x >>  4;
        x ^= x >>  2;
        x ^= x >>  1;
        return x & 1;
}

int main()
{
        int test_0 = 0x11011101;

        if(odd_ones(test_0)) {
            printf("x contains an odd number of 1s.\n");
        } else {
            printf("x contains an even number of 1s.\n");
        }
        return 0;
}
```

2.66

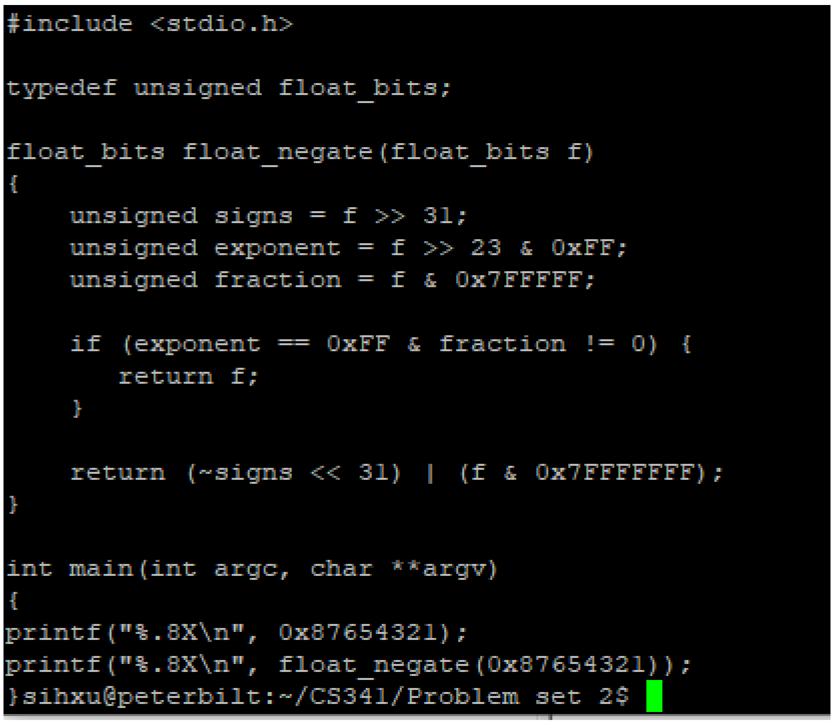


```c
#include <stdio.h>

int leftmost_one(unsigned x)
{
        x |= x >> 16;
        x |= x >>  8;
        x |= x >>  4;
        x |= x >>  2;
        x |= x >>  1;
        return x ^ (x >> 1);
}

int main(void)
{


        int test_hex_1 = 0xFF00;
        printf("%x\n",leftmost_one(test_hex_1));

        int test_hex_2 = 0x6600;
        printf("%x\n",leftmost_one(test_hex_2));

        return 0;
}
```

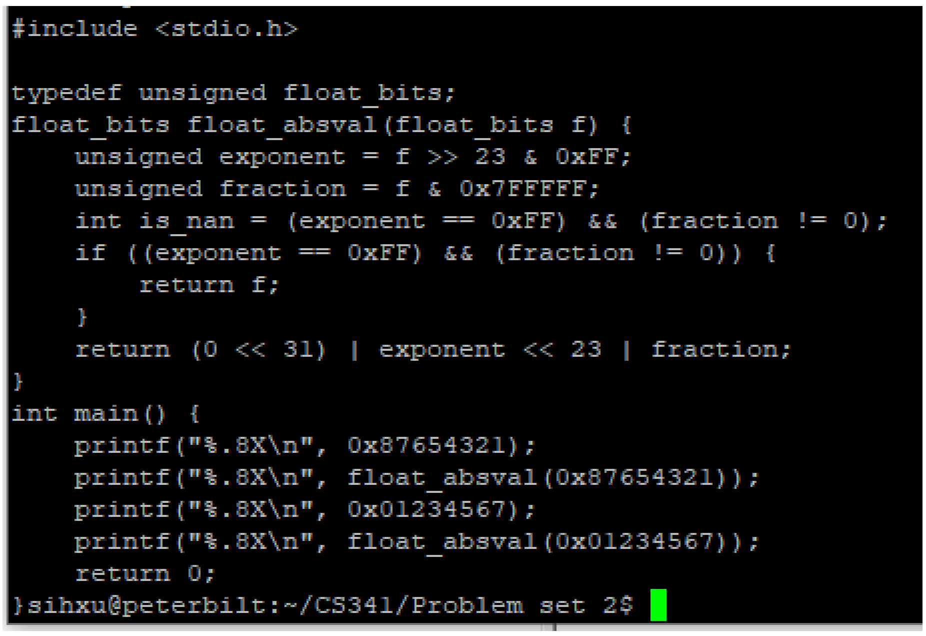
2.68


```
#include <stdio.h>
#include <assert.h>

int lower_one_mask(int n)
{
        int mask = (2 << (n - 1)) - 1;
        return mask;
}

int main()
{
        assert(lower_one_mask(6) == 0x3F);
        assert(lower_one_mask(17) == 0x1FFFF);

        printf("Successful\n");
        return 0;
}sihxu@peterbilt:~/CS341/Problem set 2$
```

2.92


```
#include <stdio.h>

typedef unsigned float_bits;

float_bits float_negate(float_bits f)
{
    unsigned signs = f >> 31;
    unsigned exponent = f >> 23 & 0xFF;
    unsigned fraction = f & 0x7FFFFF;

    if (exponent == 0xFF & fraction != 0) {
       return f;
    }

    return (~signs << 31) | (f & 0x7FFFFFFF);
}

int main(int argc, char **argv)
{
printf("%.8X\n", 0x87654321);
printf("%.8X\n", float_negate(0x87654321));
}sihxu@peterbilt:~/CS341/Problem set 2$
```

2.93



```
#include <stdio.h>

typedef unsigned float_bits;
float_bits float_absval(float_bits f) {
    unsigned exponent = f >> 23 & 0xFF;
    unsigned fraction = f & 0x7FFFFF;
    int is_nan = (exponent == 0xFF) && (fraction != 0);
    if ((exponent == 0xFF) && (fraction != 0)) {
        return f;
    }
    return (0 << 31) | exponent << 23 | fraction;
}
int main() {
    printf("%.8X\n", 0x87654321);
    printf("%.8X\n", float_absval(0x87654321));
    printf("%.8X\n", 0x01234567);
    printf("%.8X\n", float_absval(0x01234567));
    return 0;
}sihxu@peterbilt:~/CS341/Problem set 2$
```