Exercise 4

1. Pointer

What's the meaning of the following declarations?

a. `char **argv ;`
   pointer argv to pointer to char

b. `int (*daytab)[13]`
   pointer daytab to array[13] of int

c. `int (*comp)()`
   pointer comp to function returning int

d. `char (*(*x())[])()`
   x is function return pointer of array[] of pointer to function returning char

e. `char(*(*x[3])())[5]`
   array[3] of pointer x to function return pointer to char[5]

2. Buffer Overflow

One of TAs of ICS wrote a buggy function. The following C code and (part of) its assembly code are executed on a **64-bit little endian** machine.

```
int verify() {
    char password[16];
    gets(password);
    return check_match_in_database(password);
}
```

```
pushq %rbp
movq %rsp, %rbp
subq $16, %rsp
leal -16(%rbp), %rax
movq %rax, rdi
call _gets
...
```

a. In the normal process, *check_match_in_database* function will check the password and call *verify_ok* function if the password is right. But now we do not know the right password. Assume we know that the address of function *verify_ok* is 0x4005a8. Construct an input to *gets* function to let the program return to *verify_ok*. NOTE: You just need to specify the key bytes and their positions.

password[24, 25, 26, 27, 28, 29, 30, 31]=0xa8, 0x05, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00

b. Now we use string "000000001111222233330000432100000000000000" to feed the gets function. What will happen to the program?

It will return to 0x3030303031323334

3. Floating point

The following figure shows the floating-point representation called Float12, it's same as the IEEE floating-point format except for the length.

S	Exp(4bits)	Fract(7bits)

1. Fill the blanks with proper values.

1) Normalized: $(-1)^S$ * (1.Fract) * $2^{(Exp-bias)}$, where bias = __7____;

2) -∞ = _1111 1000 0000___ (in binary form);

3) Smallest Negative Denormalized Value (in binary form): ___1000 0111 1111__, and it's value in form of a * 2^b (a and b are both integers)_-127*2^(-13)_;

4) Largest negative Normalized value (in binary form)_1000 1000 0000____, and it's value in form of a * 2^b (a and b are both integers) __-1*2^-6__;

2. Convert $(-0.375)_{10}$ into the Float12 representation (in binary).

0.375=1.5*2^-2

E=-2 Exp = -2+7 = 5

1 0101 1000000

3. Assume we use IEEE round-to-even mode to do the approximation. Now a, b are both Float12 and are represented in hex. Compute a+b and fill in the following table.

	binary	E	Signed aligned M	Sum of M & result
A=0x5e3	0101 1110 0011	4	1.1100011	10.01111011 (sum)
B=0x535	0101 0011 0101	3	0.10110101	0110 0001 1111 (res)
A=0x552	0101 0101 0010	3	1.1010010	0.11101111
B=0xcb5	1100 1011 0101	2	-0.10110101	0100 1110 1111
A=0x6a9	0110 1010 1001	6	1.0101001	1.101111110
B=0x5da	0101 1101 1010	4	0.011011010	0110 1110 0000
A=0x093	0000 1001 0011	-6	1.0010011	1.1101101
B=0x05a	0000 0101 1010	-6	0.1011010	0000 1110 1101