Exercise 4

1. Pointer

What's the meaning of the following declarations?

a. `char **argv ;`

b. `int (*daytab)[13]`

c. `int (*comp)()`

d. `char (*(*x())[])()`

e. `char(*(*x[3])())[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. The ASCII of '0' is 48.

b. Now we use string "00000000111122223333000043210000000000000" to feed the gets function. What will happen to the program?

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: $(-1)^S$ * (_____) * $2^{(\_\_\_\_)}$, where bias = _____;
   2) -∞ = _____ (in binary form);
   3) Smallest Negative Denormalized Value (in binary form): _____, and it's value in form of a * 2^b (a and b are both integers) _____;
   4) Largest negative Normalized value (in binary form) _____, and it's value in form of a * 2^b (a and b are both integers)_____;
2. Convert $(-0.375)_{10}$ into the Float12 representation (in binary).

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				
B=0xcb5				
A=0x6a9				
B=0x5da				
A=0x093				
B=0x05a				