ICS Homework 11

1. Pointers and array

Answer following questions and explain why. Assume we use x86-64 machines.

1. Is the value of `&(a[1])` equals to value of `(b+1)`?

```
int a[2]; char *b = a;
```

No, sizeof(int) is 4, sizeof(char) is 1.

2. Is the value of `&(a[1])` equals to value of `(b+1)`?

```
int a[2]; char **b = a;
```

No, sizeof(int) is 4, sizeof(char) * is 8.

3. Is the value of `&(a[1])` equals to value of `(b+1)`?

```
int *a[2]; char **b = a;
```

Yes, both a and b are pointer to pointers.

4. Is the value of `&(a[1])` equals to value of `(b+1)?`

```
int a[2]; char (*b)[2][2] = a;
```

Yes, b is a pointer to a 2D array, and the size of this 2D array is 4 bytes.

5. Is the value of `&(a[1])` equals to value of `(b+1)`?

```
int a[2]; char (**b)[2][2] = a;
```

No, b is a pointer points to a pointer to a 2D array, so b+1 is 8 byte-advanced than b.

6. What is `a`?

```
int *(*a[3])(int *, int);
```

An Array with 3 elements points to a function with two parameters (int * and int) returning int pointer.

2. Buffer Overflow

The following C code and assembly code are executed on a **64-bit little endian** machine. It uses **`gets()`** functions in section 3.10.3 on CSAPP.

```
void buggy(){
    char buf[0x10];
    gets(buf);
}
int main(){
    buggy();
    return 0;
}
```

```
 00000000004004e6 <buggy>:
   4004e6:       55                      push   %rbp
   4004e7:       48 89 e5                mov    %rsp,%rbp
   4004ea:       48 83 ec 10             sub    $0x10,%rsp
   4004ee:       48 8d 45 f0             lea    -0x10(%rbp),%rax
   4004f2:       48 89 c7                mov    %rax,%rdi
   4004f5:       e8 17 00 00 00          callq  400511 <gets>
   4004fa:       c9                      leaveq
   4004fb:       c3                      retq

 00000000004004fc <main>:
   4004fc:       55                      push   %rbp
   4004fd:       48 89 e5                mov    %rsp,%rbp
   400500:       b8 00 00 00 00          mov    $0x0,%eax
   400505:       e8 dc ff ff ff          callq  4004e6 <buggy>
   40050a:       b8 00 00 00 00          mov    $0x0,%eax
   40050f:       5d                      pop    %rbp
   400510:       c3                      retq
```

Give the corresponding return address of function **`buggy()`** to each return address. (NOTE: the ASCII number of ‘0’ is 48.)

a. `""` `0x40050a`

b. `"0123456789"` `0x40050a`

c. `"01234567890123456789"` `0x40050a`

d. `"012345678901234567890123"` `0x400500`

e. `"0123456789012345678901234567890123456789"` `0x393837363534`