Homework 7

1. Buffer Overflow

One of TAs of ICS wrote a buggy program. The following C code and assembly code are executed on a **64-bit little endian** machine. He used **`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
```

Now the TA uses different strings to feed the **`gets()`** in **`buggy()`**. Give the corresponding return address of function **`buggy()`** to each return address. (NOTE: the ASCII number of ‘0’ is 48.

a. “”

b. “0123456789”

c. “01234567890123456789”

d. “012345678901234567890123”

e. “012345678901234567890123456789”

2. Floating point

Consider a 16-bit floating-point representation based on the IEEE floating-point format, with 1 sign bit, 5 exp bits, 10 frac bits, called Float16.

(1) Fill in the following table. Represent M in the form x or x/y where x is an integer and y is an integral power of 2, and represent Value in the form a or a * 2^b where a and b are integers.

Description	Hex	M	E	Value
-0				--
Largest negative Normalized value				
+∞		--	--	--
Largest Denomalized value				
$(11.375)_{10}$				
Number with hex representation 0x4BF7	0x4BF7			

(2) Assume we use IEEE round-to-even mode to do the approximation. Now a, b are both Float16, with a = 0x4663 and b = 0x394c represented in hex. Compute a+b and represent the answer in hex.

(3) Using Float16, what’s the difference between 2^15 + 0.5 – 2^15 and 2^15 – 2^15 + 0.5? Calculate them to explain why.