Weaver Fall 2020 — CS 161 Computer Security — Midterm

For questions with **circular bubbles**, you may select exactly *one* choice on Gradescope.

◯ Unselected option

● Only one selected option

For questions with **square checkboxes**, you may select *one* or more choices on Gradescope.

■ You can select

■ multiple squares

For questions with a **large box**, you need write and label your answer in the corresponding text box on Gradescope.

You have 110 minutes, plus a 10 minute buffer, for a total of 120 minutes. There are 7 questions of varying credit (120 points total).

The exam is open note. You can use an unlimited number of handwritten cheat sheets, but you must work alone.

Clarifications will be posted at https://cs161.org/clarifications.

Q1 *MANDATORY – Honor Code* (2 points)

Read the following honor code and type your name on Gradescope. *Failure to do so will result in a grade of 0 for this exam.*

> I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam.

This is the end of Q1. Proceed to Q2 on your answer sheet.

Q2 *True/false* **(30 points)**

Each true/false is worth 2 points.

Q2.1 TRUE or FALSE: Pointer authentication prevents all buffer overflow attacks.

◯ TRUE ◯ FALSE

Q2.2 TRUE or FALSE: The RET2ESP (Return to ESP) exploit from Project 1, Question 6 does **not** require knowing the **absolute** address of the shellcode when crafting the exploit.

◯ TRUE ◯ FALSE

Q2.3 TRUE or FALSE: The function $f(x) = 1$ is a one-way function, since we can't go from 1 to our original value of x.

◯ TRUE ◯ FALSE

Q2.4 TRUE or FALSE: EvanBot designs custom buffer overflow protection that blocks all writes to RIP's and SFP's. This is a successful defense against all buffer overflow attacks.

◯ TRUE ◯ FALSE

Q2.5 TRUE or FALSE: Using `fgets(buf, size, ...)` instead of `gets(buf)` always prevents an attacker from overflowing `buf`.

◯ TRUE ◯ FALSE

Q2.6 TRUE or FALSE: Diffie-Hellman is a protocol for sending messages confidentially between two people who don't share a key.

◯ TRUE ◯ FALSE

Q2.7 TRUE or FALSE: The El Gamal protocol from lecture guarantees integrity.

◯ TRUE ◯ FALSE

Q2.8 TRUE or FALSE: When using CBC mode, we need to pad messages because the block cipher takes a fixed-length input.

◯ TRUE ◯ FALSE

Q2.9 TRUE or FALSE: Kerckhoffs's principle assumes that everything about a cryptographic system, including the key, is public knowledge.

◯ TRUE ◯ FALSE

Q2.10 TRUE or FALSE: Slower hashes are useful for password hashing.

◯ TRUE ◯ FALSE

Q2.11 TRUE or FALSE: In a digital signature scheme, the verifying key is private, and the signing key is public.

○ True ○ False

Q2.12 True or False: A 64-bit stack canary on a 64-bit processor provides more protection than a 32-bit stack canary on a 32-bit processor.

○ True ○ False

Q2.13 True or False: Security is economics, so you should generally not use a $100 lock to secure a $10 product.

○ True ○ False

Q2.14 True or False: The confidentiality of El Gamal is compromised if r, the random value chosen for each message sent, is public.

○ True ○ False

Q2.15 True or False: RSA encryption without padding is IND-CPA secure.

○ True ○ False

This is the end of Q2. Proceed to Q3 on your answer sheet.

Q3 *MAC Madness* **(18 points)**

Evan wants to store a list of every CS161 student's firstname and lastname, but he is afraid Mallory will tamper with his list.

Evan is considering adding a cryptographic value to each record to ensure its integrity. For each scheme, determine what Mallory can do without being detected.

Assume MAC is a secure MAC, H is a cryptographic hash, and Mallory does not know Evan's secret key k. Assume that firstname and lastname are all lowercase and alphabetic (no numbers or special characters), and concatenation does not add any delimiter (e.g. a space or tab), so $\mathsf{nick}\|\mathsf{weaver} = \mathsf{nickweaver}$.

Clarifications during the exam: Bob is storing the names with the cryptographic value in the database. Duplicate records are not allowed. Mallory can change anything in the database. "A value of her choosing" means any arbitrary value.

Q3.1 (3 points) $\mathsf{H}(\mathsf{firstname}\|\mathsf{lastname})$

- ◯ (A) Mallory can modify a record to be a value of her choosing
- ◯ (B) Mallory can modify a record to be a specific value (not necessarily of her choosing)
- ◯ (C) Mallory cannot modify a record without being detected
- ◯ (D) ——
- ◯ (E) ——
- ◯ (F) ——

Q3.2 (3 points) $\mathsf{MAC}(k, \mathsf{firstname}\|\mathsf{lastname})$

Hint: Can you think of two different records that would have the same MAC?

- ◯ (G) Mallory can modify a record to be a value of her choosing
- ◯ (H) Mallory can modify a record to be a specific value (not necessarily of her choosing)
- ◯ (I) Mallory cannot modify a record without being detected
- ◯ (J) ——
- ◯ (K) ——
- ◯ (L) ——

Q3.3 (3 points) $\mathsf{MAC}(k, \mathsf{firstname}\|\texttt{"-"}\|\mathsf{lastname})$, where `"-"` is a hyphen character.

- ◯ (A) Mallory can modify a record to be a value of her choosing
- ◯ (B) Mallory can modify a record to be a specific value (not necessarily of her choosing)
- ◯ (C) Mallory cannot modify a record without being detected

◯ (D) —

◯ (E) —

◯ (F) —

Q3.4 (3 points) $\mathsf{MAC}(k, \mathsf{H}(\mathsf{firstname})\|\mathsf{H}(\mathsf{lastname}))$

◯ (G) Mallory can modify a record to be a value of her choosing

◯ (H) Mallory can modify a record to be a specific value (not necessarily of her choosing)

◯ (I) Mallory cannot modify a record without being detected

◯ (J) —

◯ (K) —

◯ (L) —

Q3.5 (3 points) $\mathsf{MAC}(k, \mathsf{firstname})\|\mathsf{MAC}(k, \mathsf{lastname})$

◯ (A) Mallory can modify a record to be a value of her choosing

◯ (B) Mallory can modify a record to be a specific value (not necessarily of her choosing)

◯ (C) Mallory cannot modify a record without being detected

◯ (D) —

◯ (E) —

◯ (F) —

Q3.6 (3 points) Which of Evan's schemes guarantee confidentiality on his records?

◯ (G) All 5 schemes

◯ (H) Only the schemes with a MAC

◯ (I) Only the schemes with a hash

◯ (J) None of the schemes

◯ (K) —

◯ (L) —

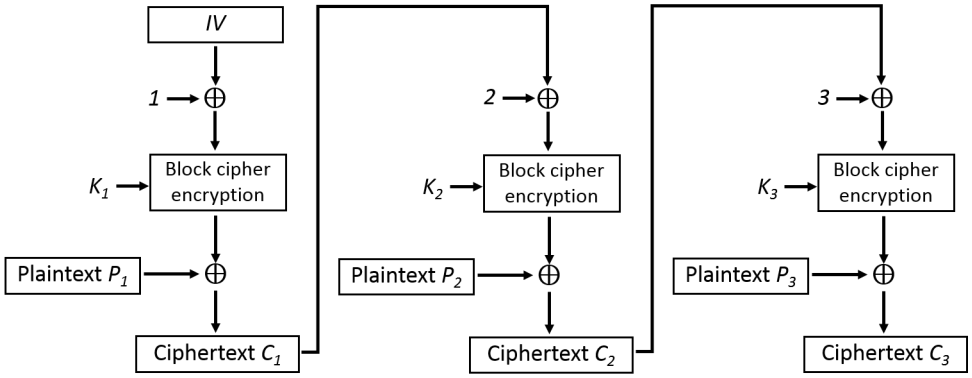
This is the end of Q3. Proceed to Q4 on your answer sheet.

Q4 *Socially Distanced Cipher* **(18 points)**

Bob and Alice want to plan a social distancing picnic, but don't want to invite Eve because she hasn't been wearing a mask in public. They decide to send messages using a new block cipher chaining mode, AES-SDC (Socially Distanced Cipher). Note that AES-SDC requires a different key for each block of the message.

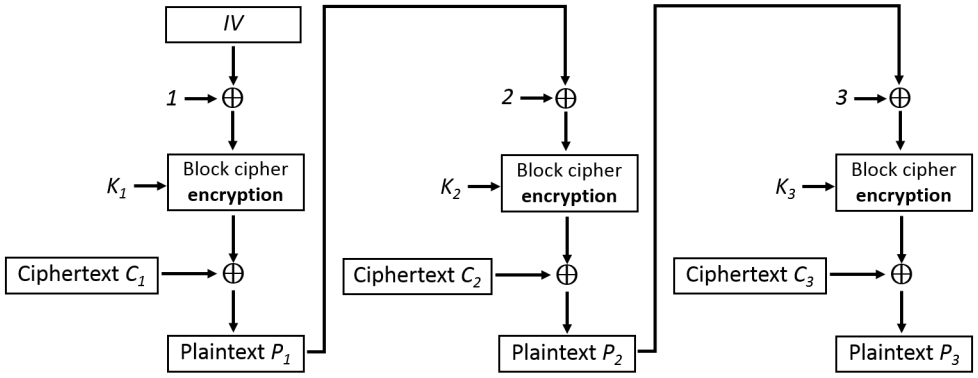
$$C_0 = IV$$

$$C_i = E_{k_i}(C_{i-1} \oplus i) \oplus P_i$$

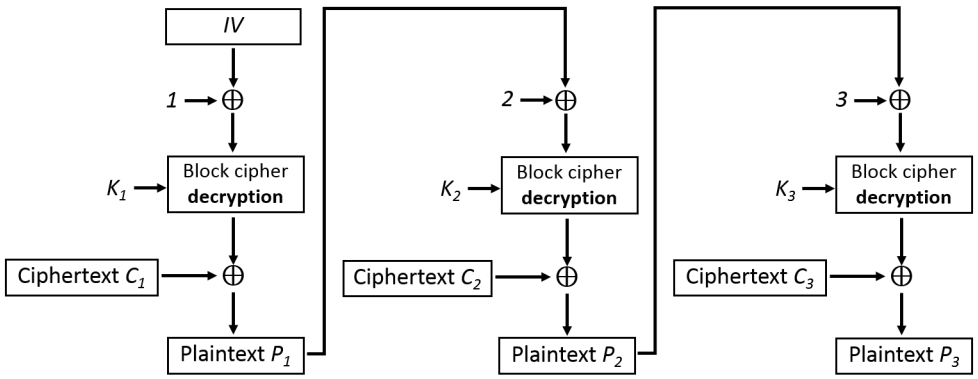


Q4.1 (3 points) Which of the following is the correct decryption expression/diagram for AES-SDC?

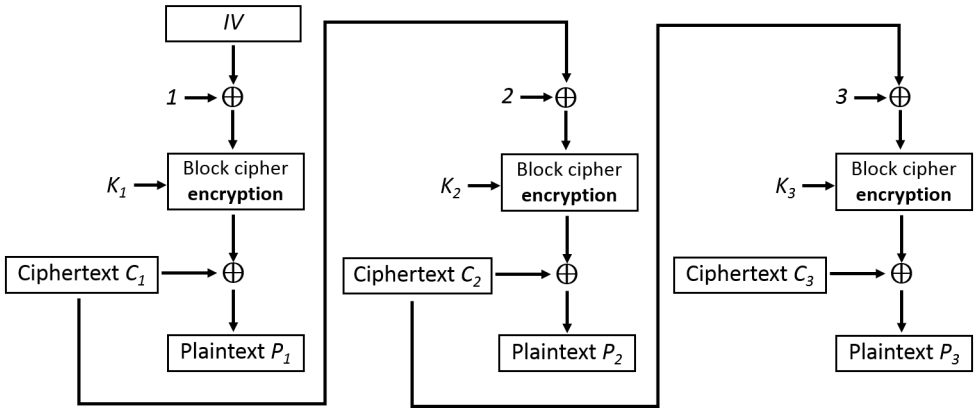
◯ (A) $P_i = E_{k_i}(P_{i-1} \oplus i) \oplus C_i$



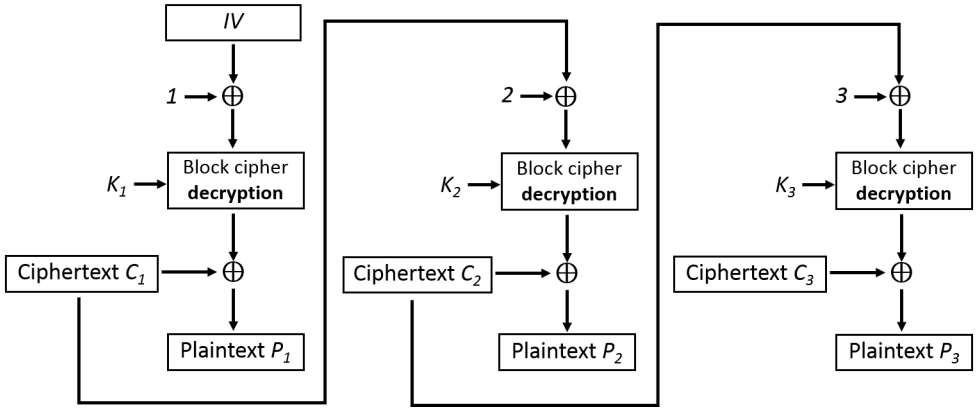
◯ (B) $P_i = D_{k_i}(P_{i-1} \oplus i) \oplus C_i$



◯ (C) $P_i = E_{k_i}(C_{i-1} \oplus i) \oplus C_i$



◯ (D) $P_i = D_{k_i}(C_{i-1} \oplus i) \oplus C_i$



◯ (E) ——

◯ (F) ——

Q4.2 (3 points) Select all true statements about this encryption scheme.

Hint: The cipher mode you saw in Homework 2, $C_i = E_k(C_{i-1}) \oplus P_i$, is IND-CPA secure.

☐ (G) Encryption can be parallelized

☐ (H) Decryption can be parallelized

☐ (I) It is IND-CPA secure

☐ (J) None of the above

☐ (K) ——

☐ (L) ——

Suppose Alice loses some of her shared keys with Bob. Alice wants to encrypt an n-block message using AES-SDC. For each scenario below, determine which blocks Alice can still encrypt.

Q4.3 (3 points) Alice has all the keys except k_4 and k_5.

◯ (A) Alice can encrypt all parts of her message except P_4 and P_5

◯ (B) Alice can encrypt P_1, P_2 and P_3 only.

◯ (C) Alice can encrypt the entire message

◯ (D) Alice cannot encrypt any block of the message

◯ (E) None of the above

◯ (F) ——

Now, suppose Alice now has all the keys, and Alice sends a n-block message to Bob. Eve learns some keys and some blocks of ciphertext. For each scenario below, determine which blocks Eve can decrypt.

Clarification during exam: Eve knows the index of all keys and ciphertext blocks that she learns.

Q4.4 (3 points) Eve learns the IV, ciphertext blocks C_5 and C_6, and key k_5.

◯ (G) Eve can decrypt C_5 only

◯ (H) Eve can decrypt C_5 and C_6 only

◯ (I) Eve can decrypt all messages intercepted

◯ (J) Eve cannot decrypt any intercepted messages

◯ (K) None of the above

◯ (L) ——

Q4.5 (3 points) Eve learns the IV, ciphertext blocks C_2, C_3, and C_5, and keys k_2, k_3, and k_5.

◯ (A) Eve can decrypt C_3 and C_5 only

◯ (B) Eve can decrypt C_2, C_3, C_5 only

◯ (C) Eve can decrypt C_2, C_3, C_4, C_5 only

◯ (D) Eve can decrypt C_3 only

◯ (E) Eve cannot decrypt any intercepted messages

◯ (F) None of the above

Q4.6 (3 points) Bob receives all the keys and ciphertext blocks C_1 through C_n, but C_3 is corrupted. Which plaintext blocks can Bob successfully decrypt?

Clarification during exam: "Bob receives all the keys and ciphertext blocks C_1 through C_n" should be "ciphertext blocks C_0 through C_n."

◯ (G) Bob can successfully decrypt all blocks except C_3

◯ (H) Bob can successfully decrypt all blocks except C_4

◯ (I) Bob can successfully decrypt all blocks except C_1, C_2, C_3

◯ (J) Bob can successfully decrypt all blocks except C_3 and C_4

◯ (K) Bob cannot successfully decrypt any of the blocks

◯ (L) None of the above

This is the end of Q4. Proceed to Q5 on your answer sheet.

Q5 *Hacked EvanBot* (16 points)

Hacked EvanBot is running code to violate students' privacy, and it's up to you to disable it before it's too late!

```
1  #include <stdio.h>
2
3  void spy_on_students(void) {
4      char buffer[16];
5      fread(buffer, 1, 24, stdin);
6  }
7
8  int main() {
9      spy_on_students();
10     return 0;
11 }
```

The shutdown code for Hacked EvanBot is located at address `0xdeadbeef`, but there's just one problem—Bot has learned a new memory safety defense. Before returning from a function, it will check that its saved return address (rip) is not `0xdeadbeef`, and throw an error if the rip is `0xdeadbeef`.

Clarification during exam: Assume little-endian x86 for all questions.

Assume all x86 instructions are 8 bytes long. Assume all compiler optimizations and buffer overflow defenses are disabled.

The address of `buffer` is `0xbffff110`.

Q5.1 (3 points) In the next 3 subparts, you'll supply a malicious input to the `fread` call at line 5 that causes the program to execute instructions at `0xdeadbeef`, *without* overwriting the rip with the value `0xdeadbeef`.

The first part of your input should be a single assembly instruction. What is the instruction? x86 pseudocode or a brief description of what the instruction should do (5 words max) is fine.

Q5.2 (3 points) The second part of your input should be some garbage bytes. How many garbage bytes do you need to write?

◯ (G) 0 ◯ (H) 4 ◯ (I) 8 ◯ (J) 12 ◯ (K) 16 ◯ (L) ——

Q5.3 (3 points) What are the last 4 bytes of your input? Write your answer in Project 1 Python syntax, e.g. `\x12\x34\x56\x78`.

Q5.4 (3 points) When does your exploit start executing instructions at `0xdeadbeef`?

◯ (G) Immediately when the program starts

◯ (H) When the `main` function returns

◯ (I) When the `spy_on_students` function returns

◯ (J) When the `fread` function returns

◯ (K) ——

◯ (L) ——

Q5.5 (4 points) Which of the following defenses would stop your exploit from the previous parts?

☐ (A) Non-executable pages (also called DEP, W^X, and the NX bit)

☐ (B) Stack canaries

☐ (C) ASLR

☐ (D) Rewrite the code in a memory-safe language

☐ (E) None of the above

☐ (F) ——

This is the end of Q5. Proceed to Q6 on your answer sheet.

Q6 *Chegg* **(17 points)**

Engineers at Chegg are analyzing different password management techniques. Unfortunately, the engineers at Chegg used Chegg to make it through CS 161, so they don't remember anything they learned!

Q6.1 (3 points) Suppose there is an offline attacker (with access to the hashed passwords file) and an online attacker (without access to the hashed passwords file). Chegg implements a CAPTCHA on its login page. Which attacker(s) does the CAPTCHA prevent from performing a dictionary attack?

Clarification during exam: "prevent from performing a dictionary attack" means make the attack significantly more expensive.

◯ (A) The offline attacker only

◯ (B) The online attacker only

◯ (C) Both attackers

◯ (D) Neither attacker

◯ (E) ——

◯ (F) ——

Q6.2 (3 points) Instead of salting each password hash, Chegg engineers XOR the hashed password with the account creation timestamp and store the XOR'd password hash with the timestamp in their database.

True or false: This successfully prevents an offline attacker from performing a dictionary attack.

Clarification during exam: The timestamp and the XOR'd password hash are both stored in the database. The offline attacker has the entire database.

◯ (G) True ◯ (H) False ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

Q6.3 (4 points) One of Chegg's competitors, Course Hero, has been compromised, and all of their user accounts and passwords have been leaked in plaintext. Select all defenses that Chegg could use to protect students who use the same password for Chegg and Course Hero.

☐ (A) Use a slow hash function

☐ (B) Include a salt in the password hash e.g. store a tuple of $(\mathsf{salt}, \mathsf{H}(\mathsf{salt}\|\mathsf{password}))$

☐ (C) Require every login attempt to also provide a random code sent by a secure SMS to the registered user's phone (a secure second factor)

☐ (D) During the account creation phase, require every password to end with `-CHEGG`

☐ (E) None of the above

☐ (F) ——

Chegg uses a certificate chain in order to verify tutors. When tutors post responses, they attach a digital signature of their response along with their certificate. Students can verify the authenticity of a response by verifying the certificate and using the public key in the certificate to verify the signature.

The certificate chain is below. Assume that the Chegg Root Certificate Authority (CA) is hardcoded into students' browsers.

1. Identity: Director of Chegg Recruiting (Verified by Chegg Root CA)
2. Identity: Campus Chegg Recruiter (Verified by Director of Chegg Recruiting)
3. Identity: Authorized Tutor (Verified by Campus Chegg Recruiter)

Q6.4 (4 points) EvanBot is not a valid tutor, but wants to create a fake tutor response with a valid signature. Which of these attacks would allow Bot to accomplish this?

☐ (G) Steal the public key of the Campus Chegg Recruiter

☐ (H) Steal the private key of the Director of Chegg Recruiting

☐ (I) Steal the private key of the Chegg Root CA

☐ (J) Steal the certificate of an authorized tutor

☐ (K) None of the above

☐ (L) ——

Q6.5 (3 points) EvanBot gains access to the private key of Dave, who is an authorized tutor. Which of the following can EvanBot do?

☐ (A) Post a valid response as Nick, an existing tutor

☐ (B) Post a valid response as Dave

☐ (C) Create and sign a certificate for Raluca, a new tutor

☐ (D) None of the above

☐ (E) ——

☐ (F) ——

This is the end of Q6. Proceed to Q7 on your answer sheet.

Q7 *Stack Exchange* **(19 points)**

Consider the following vulnerable C code:

```c
#include <byteswap.h>
#include <inttypes.h>
#include <stdio.h>

void prepare_input(void) {
    char buffer[64];
    int64_t *ptr;

    printf("What is the buffer?\n");
    fread(buffer, 1, 68, stdin);

    printf("What is the pointer?\n");
    fread(&ptr, 1, sizeof(uint64_t *), stdin);

    if (ptr < buffer || ptr >= buffer + 68) {
        printf("Pointer is outside buffer!");
        return;
    }

    /* Reverse 8 bytes of memory at the address ptr */
    *ptr = bswap_64(*ptr);
}

int main(void) {
    prepare_input();
    return 0;
}
```

The `bswap_64` function takes in 8 bytes and returns the 8 bytes in reverse order.

Assume that the code is run on a 32-bit system, no memory safety defenses are enabled, and there are no exception handlers, saved registers, or compiler padding.

Q7.1 (3 points) Fill in the numbered blanks on the following stack diagram for `prepare_input`.

1	(0xbffff494)
2	(0xbffff490)
3	(0xbffff450)
4	(0xbffff44c)

◯ (A) 1 = `sfp`, 2 = `rip`, 3 = `buffer`, 4 = `ptr`

◯ (B) 1 = `sfp`, 2 = `rip`, 3 = `ptr`, 4 = `buffer`

◯ (C) 1 = `rip`, 2 = `sfp`, 3 = `buffer`, 4 = `ptr`

◯ (D) 1 = `rip`, 2 = `sfp`, 3 = `ptr`, 4 = `buffer`

◯ (E) —

◯ (F) —

Q7.2 (4 points) Which of these values on the stack can the attacker write to at lines 10 and 13? Select all that apply.

☐ (G) `buffer`

☐ (H) `ptr`

☐ (I) `sfp`

☐ (J) `rip`

☐ (K) None of the above

☐ (L) ——

Q7.3 (3 points) Give an input that would cause this program to execute shellcode. At line 10, first input these bytes:

◯ (A) 64-byte shellcode

◯ (B) `\xbf\xff\xf4\x4c`

◯ (C) `\x4c\xf4\xff\xbf`

◯ (D) `\xbf\xff\xf4\x50`

◯ (E) `\x50\xf4\xff\xbf`

◯ (F) ——

Q7.4 (3 points) Then input these bytes:

◯ (G) 64-byte shellcode

◯ (H) `\xbf\xff\xf4\x4c`

◯ (I) `\x4c\xf4\xff\xbf`

◯ (J) `\xbf\xff\xf4\x50`

◯ (K) `\x50\xf4\xff\xbf`

◯ (L) ——

Q7.5 (3 points) At line 13, input these bytes:

◯ (A) `\xbf\xff\xf4\x50`

◯ (B) `\x50\xf4\xff\xbf`

◯ (C) `\xbf\xff\xf4\x90`

◯ (D) `\x90\xf4\xff\xbf`

◯ (E) `\xbf\xff\xf4\x94`

◯ (F) `\x94\xf4\xff\xbf`

Q7.6 (3 points) Suppose you replace `68` with `64` at line 10 and line 15. Is this modified code memory-safe?

◯ (G) Yes ◯ (H) No ◯ (I) —— ◯ (J) —— ◯ (K) —— ◯ (L) ——

This is the end of Q7. You have reached the end of the exam.

C Function Definitions

```
bswap_64(x);
```

Returns a value in which the order of the bytes in its 8-byte argument is reversed.

```
char *fgets(char *s, int size, FILE *stream);
```

fgets() reads in at most one less than size characters from stream and stores them into the buffer pointed to by s. Reading stops after an EOF or a newline. If a newline is read, it is stored into the buffer. A terminating null byte ('\0') is stored after the last character in the buffer.

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

The function fread() reads nmemb items of data, each size bytes long, from the stream pointed to by stream, storing them at the location given by ptr.

Note that fread() does not add a null byte after input.