CS 161 Spring 2023 — Introduction to Computer Security — Midterm

PRINT your name: ____________________, ____________________
(last) (first)

PRINT your student ID: ______________________________

There are 7 questions of varying credit (150 points total).

Question:	1	2	3	4	5	6	7	Total
Points:	2	28	23	23	24	28	22	150

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (completely unfilled)
- ● Only one selected option (completely filled)

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

- ☐ You can select
- ■ multiple squares (completely filled)

Pre-exam activity (not graded, just for fun):
Look, a ~~shooting star~~ deorbited satellite! What will you wish for?

____

Q1 *Honor Code* (2 points)

Read the following honor code and sign your name.

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.

SIGN your name:

____

Q2 *True/False* **(28 points)**

Each true/false is worth 2 points.

For the next 2 subparts: Tired of biased referees, the Caltopia Sports Association (CSA) is now using EvanBot to officiate the upcoming games.

Q2.1 All changes to EvanBot's programming are logged in a secure file for auditing.

TRUE or FALSE: This is an example of detecting if you can't prevent.

○ TRUE ○ FALSE

Q2.2 CSA officials and team coaches are given full read/write access to EvanBot's programming to audit and ensure fairness.

TRUE or FALSE: This is an example of separation of responsibility.

○ TRUE ○ FALSE

For the next 2 subparts: You run `x/4wx buf` and receive the following GDB output:

```
0xffff1244: 0x00ab0000 0x00000000 0x00000033 0xff000000
```

Q2.3 TRUE or FALSE: Byte `0xab` appears at address `0xffff1246` in memory.

○ TRUE ○ FALSE

Q2.4 Suppose `buf` is a local variable defined in function `foo`.

TRUE or FALSE: It is possible that the address of `foo`'s RIP is `0xffff1260`.

○ TRUE ○ FALSE

Q2.5 Recall the off-by-one exploit from the project.

TRUE or FALSE: Without any modifications, this exploit will crash the program if stack canaries are enabled.

○ TRUE ○ FALSE

Q2.6 An attacker writes `"rm -rf /"` into memory and calls the `system` function with this string as an argument.

TRUE or FALSE: Non-executable pages will stop this attack because `"rm -rf /"` was written to a non-executable part of memory.

○ TRUE ○ FALSE

For the next 2 subparts: Alice and Bob share a symmetric key K not known to anyone else. Alice sends these two values to Bob:

$$M \text{ and } \mathsf{SHA\text{-}2}(M \| K)$$

Q2.7 TRUE or FALSE: A MITM can modify the message M to be some arbitrary amount of their choosing, without being detected.

○ TRUE ○ FALSE

Q2.8 TRUE or FALSE: A MITM can modify the message M to be some different amount (not necessarily of their choosing), without being detected.

○ TRUE ○ FALSE

For the next 3 subparts: You perform a Diffie-Hellman key exchange with EvanBot over an insecure channel. You use a randomly-generated secret a, but EvanBot uses $b = 0$.

Q2.9 TRUE or FALSE: You and EvanBot will derive the same shared secret.

○ TRUE ○ FALSE

Q2.10 TRUE or FALSE: An eavesdropper on the insecure channel can derive the shared secret.

○ TRUE ○ FALSE

Q2.11 TRUE or FALSE: An eavesdropper on the insecure channel can learn your shared secret a.

○ TRUE ○ FALSE

Q2.12 Consider a world where everyone uses Diffie-Hellman to exchange shared secrets, and then uses those shared secrets to communicate with only symmetric-key cryptography.

TRUE or FALSE: In this world, there would be no benefit to introducing certificates.

○ TRUE ○ FALSE

For the next 2 subparts: Consider the following certificate tree:

1. EvanBot (root of trust)
2. TAs (certificates signed by EvanBot)
3. Readers (certificates signed by a TA)

Q2.13 TRUE or FALSE: If an attacker compromises the private key of a reader, they can create a valid certificate endorsing the attacker as a reader.

○ TRUE ○ FALSE

Q2.14 TRUE or FALSE: If the TA certificates issued by EvanBot expire every 24 hours, then both the TAs and readers need to renew their certificates every 24 hours. Assume that renewing a certificate involves creating a new public/private keypair.

○ TRUE ○ FALSE

Q3 *IND-CPA and Block Ciphers: evanbotevanbotevanbotevan...* **(23 points)**

Consider a PRNG whose outputs repeat in a cycle of length 6:

$$\ldots 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, \ldots$$

Let r_i denote the ith output from the PRNG. We don't know the initial seed of this PRNG, but we do know the first output (r_0) will be an integer between 1 and 6 (inclusive). After initially seeding or reseeding, the next output has an equal probability of being any of the 6 numbers.

Consider using this PRNG to build an encryption scheme:

$$C_i = E_K(H(r_i)) \oplus M_i$$

For each block we need to encrypt, we generate the next output r_i from the PRNG, and compute C_i using the equation above, where E_K denotes AES encryption.

Here's an example of this encryption scheme. If the first PRNG output will be 1, and we encrypt a 3-block message (M_0, M_1, M_2), then the ciphertext (C_0, C_1, C_2) would be computed as follows:

$$C_0 = E_K(H(1)) \oplus M_0$$
$$C_1 = E_K(H(5)) \oplus M_1$$
$$C_2 = E_K(H(4)) \oplus M_2$$

Q3.1 (3 points) Write the decryption formula for M_i.

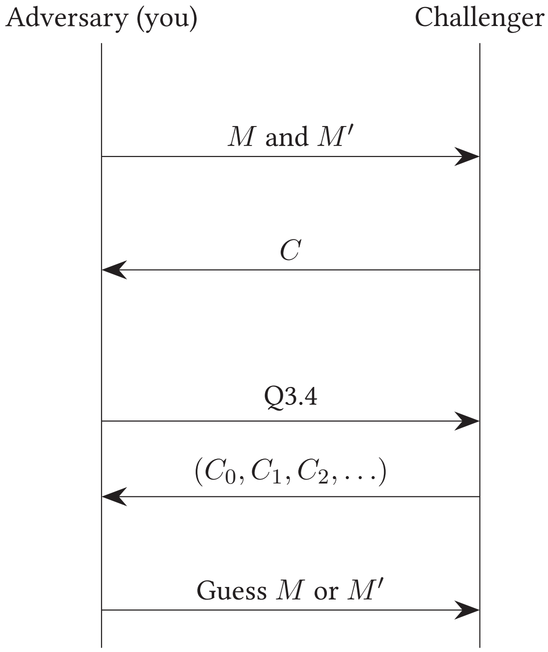
Q3.2 (4 points) Select all true statements about this scheme.

- ☐ If an attacker flips the last bit of C_i, then the last bit of P_i will also be flipped.
- ☐ If an attacker flips the last bit of C_i, then P_i will become random-looking garbage.
- ☐ If an attacker flips the last bit of C_{i-1}, then the last bit of P_i will also be flipped.
- ☐ If an attacker flips every bit of C_i, then every bit of P_i will also be flipped.
- ☐ None of the above

Q3.3 (3 points) For any two plaintext blocks M_i and M_j, which of the following conditions would cause the corresponding ciphertext blocks to be equal ($C_i = C_j$)?

- ○ $M_i \neq M_j$, and $i - j$ is a multiple of 6.
- ○ $M_i = M_j$, and $i - j$ is exactly 6.
- ○ $M_i \neq M_j$, and $i - j$ is exactly 6.
- ○ There is no way to deduce that $C_i = C_j$.
- ○ $M_i = M_j$, and $i - j$ is a multiple of 6.

To show this scheme is insecure, you want to provide a strategy that always wins the IND-CPA game.



Challenge phase:

First, the adversary (that's you!) sends two different challenge messages, $M \neq M'$, to the challenger. For your strategy, you can assume **both M and M' are one block long**.

The challenger randomly picks one of M or M' to encrypt. The resulting **one-block ciphertext** C is returned to you.

Query phase:

You send **one** multi-block query message to the challenger...

...and you receive the encryption of that query message.

Finally, you will guess whether M or M' was sent (see Q3.5).

Q3.4 (3 points) In general, you can ask the challenger to encrypt multiple query messages in the query phase. However, in your strategy, you only need to ask the challenger to encrypt **one** query message.

Which of these choices of query message would work in your strategy? Select all that apply.

(Note: 0 is a block of all zeroes.)

☐ (M, M')

☐ $(0, 0, 0, 0)$

☐ $(0, 0, 0, 0, 0, 0, 0)$

☐ (M, M, M, M, M, M)

☐ $(0, M, M')$

☐ (M', M', M', M', M', M')

Q3.5 (6 points) Describe the strategy that you will use to win the IND-CPA game.

In the query phase, the message you sent to the challenger is:

(Write one of the answers you selected in the previous subpart. All the answers you selected should work, but just pick one here to use in your strategy.)

The challenger encrypts the message you wrote in the box above and returns $(C_0, C_1, C_2, \ldots)$.

Explain how you would determine whether M or M' was encrypted, using:

- M and M' from the challenge phase,
- the encryption C from the challenge phase (C is either the encryption of M or M'), and
- the encryption of the message you wrote in the box above $(C_0, C_1, C_2, \ldots)$.

An example of how you could describe your strategy, that has nothing to do with this question: If C equals $C_1 \oplus 161$ or $C_2 \oplus C_5$, guess M. Else, guess M'.

Q3.6 (4 points) In general, what must be true about the query message (from Q3.4) in order for this strategy to work? You can answer in 10 words or fewer.

Q4 *Public-Key Cryptography: Mallory Forger* (23 points)

Alice wants to securely send a number M to the bank, to tell the bank to send M dollars to Bob. However, Mallory might try to read or tamper with M.

Assumptions:

- Both the bank and Alice have published, trusted RSA public keys denoted PK_{bank} and PK_{alice}, respectively.
- There are too many possible values of M for Mallory to try them all in a brute-force attack.
- Alice sends exactly one message to the bank.

Alice and her bank decide to use the RSA encryption scheme from class (no OAEP padding):

- $\mathsf{Enc}(PK, M) = M^e \bmod N$
- $\mathsf{Dec}(SK, C) = C^d \bmod N$

e, N are from the RSA public key PK, d is the RSA private key SK, and C is the ciphertext to be decrypted.

Scheme 1: Alice sends $\mathsf{Enc}(PK_{\text{bank}}, M)$.

Q4.1 (2 points) Can Mallory learn the value of M?

- ○ Yes
- ○ No

Q4.2 (2 points) Can Mallory change M without being detected?

- ○ Yes, Mallory can change M to any value of her choosing.
- ○ Yes, but Mallory can only change M to a specific value (not necessarily of her choosing).
- ○ No, Mallory cannot change M.

Scheme 2: Alice sends $\mathsf{Enc}(PK_{\text{bank}}, M)$ and $H(M)$.

Q4.3 (2 points) Can Mallory learn the value of M?

- ○ Yes
- ○ No

Q4.4 (2 points) Can Mallory change M without being detected?

- ○ Yes, Mallory can change M to any value of her choosing.
- ○ Yes, but Mallory can only change M to a specific value (not necessarily of her choosing).
- ○ No, Mallory cannot change M.

Alice and her bank decide to also use the **simplified** RSA signature scheme from class. This is similar to the full RSA signature scheme from class, except the message M is not being hashed:

- $\mathsf{Sign}(SK, M) = M^d \bmod N$
- $\mathsf{Verify}(PK, S, M)$: check if $S^e \equiv M \bmod N$

e, N are from the RSA public key PK, d is the RSA private key SK, and S refers to a signature output by Sign.

Scheme 3: Alice sends $C = \mathsf{Enc}(PK_{\text{bank}}, M)$ and $S = \mathsf{Sign}(SK_{\text{alice}}, M)$.

Q4.5 (5 points) Can Mallory learn the value of M?

- ○ Yes
- ○ No

Describe (in words or equations) how Mallory learns M, or explain why Mallory cannot learn M.

Q4.6 (5 points) Can Mallory change M without being detected?

- ○ Yes, Mallory can change M to any value of her choosing.
- ○ Yes, but Mallory can only change M to a specific value (not necessarily of her choosing).
- ○ No, Mallory cannot change M.

Describe (in words or equations) how Mallory changes M (and possibly S), or explain why Mallory cannot change M.

Scheme 4: Alice sends $C = \mathsf{Enc}(PK_{\text{bank}}, M)$ and $S = \mathsf{Sign}(SK_{\text{alice}}, C)$.

Q4.7 (2 points) Can Mallory learn the value of M?

- ○ Yes
- ○ No

Q4.8 (3 points) Can Mallory change M without being detected?

- ○ Yes, Mallory can change M to any value of her choosing.
- ○ Yes, but Mallory can only change M to a specific value (not necessarily of her choosing).
- ○ No, Mallory cannot change M.

Q5 *Passwords and Integrity: alice161* (24 points)

Consider a password storage server, with the following assumptions:

- There are N users who each choose from a common pool of N possible **alphanumeric** passwords.
- Usernames are unique. Passwords may not be unique.

Consider an attacker who is able to:

- Compute $O(N)$ hashes
- Read and modify what is stored in the password database
- Perform offline brute-force attacks (but not online attacks)

Scheme 1: For each user, the server stores the username and these two values:

$$\mathsf{H}(\texttt{password}) \text{ and } \mathsf{Sign}(SK, \mathsf{H}(\texttt{password}))$$

Sign is a secure digital signature scheme, and SK is a key known only by the server.

Q5.1 (3 points) In Scheme 1, is the attacker able to determine all pairs of users who share the same password?

- ○ Yes, without computing any hashes
- ○ Yes, but only by computing $O(N)$ hashes
- ○ No

Q5.2 (3 points) In Scheme 1, how many users' passwords is the attacker able to learn?

○ None ○ Only one ○ All of them

Q5.3 (3 points) In Scheme 1, what is the attacker able to do to a specific user's password?

- ○ Change the password to a specific value (not necessarily of the attacker's choosing)
- ○ Change the password to any value of the attacker's choosing
- ○ Nothing

Scheme 2: For each user, the server stores the username and

$$\mathsf{HMAC}(K, \texttt{username} \parallel \texttt{"161"} \parallel \texttt{password})$$

K is a key known only by the server.

Q5.4 (3 points) In Scheme 2, is the attacker able to determine all pairs of users who share the same password?

- ○ Yes, without computing any hashes
- ○ Yes, but only by computing $O(N)$ hashes
- ○ No

Q5.5 (3 points) In Scheme 2, how many users' passwords is the attacker able to learn?

○ None ○ Only one ○ All of them

For the rest of the question, consider this scheme:

Scheme 3: For each user, the server stores the username and these two values:

$$C = \mathsf{Enc}(K_1, \texttt{username} \parallel \texttt{"161"} \parallel \texttt{password})$$

$$\mathsf{HMAC}(K_2, C)$$

Enc is an IND-CPA secure encryption scheme. K_1 and K_2 are secret keys known only by the server.

There are two ways a user can interact with the password server:

Create user: The user provides a new `username` and a new `password`. The server computes and stores the two values above for the new user. The attacker is able to see these new values in the database.

Log in:

- The user provides their existing `username` and their `password`.
- The server first verifies the ciphertext C with the stored HMAC value.
- Then, the server decrypts C, and compares the decrypted plaintext with the string `username161password` (replacing `username` and `password` with user-provided values).
- If the decrypted plaintext matches the user-provided string, the user is logged in.

The attacker is able to perform both of these operations, and read/modify the database at all times.

Q5.6 (4 points) The database contains one user, with username `alice161`. The attacker does not know their password.

Explain how the attacker could log in as `alice161`. You may not try to log in with all possible passwords.

Hint: Try writing out what string will be encrypted for user `alice161`.

Q5.7 (5 points) The database contains one user, with username `evanbot`. The attacker does not know their password.

Explain how the attacker could log in as `evanbot`. You may not try to log in with all possible passwords.

Q6 *Format Strings: Cake Without Pan* (28 points)

For this entire question, each subpart is independent of the others.

Consider the following stack diagram after `printf(buf)` is called:

Stack

Address	Value
`0xffff1248:`	`0x12345678` ⟶ "evanbot"
`0xffff1244:`	`0xffff1234`
`0xffff1240:`	`0xdabbad04` ⟶ `0x00000041`
`0xffff123c:`	`0xdeadbeef`
`0xffff1238:`	`0xffff1250`
`0xffff1234:`	`0x1234001d` ⟶ "pancaketasty"
`0xffff1230:`	`&buf`
`0xffff122c:`	RIP of `printf`
`0xffff1228:`	SFP of `printf`

How to read this diagram: Each row of the diagram represents a value on the stack, and if relevant, the data located at that address (shown with arrows).

For example: at address `0xffff1234`, you'll find the address `0x1234001d`. At address `0x1234001d`, you'll find the null-terminated string `pancaketasty`.

Directions: For each subpart, provide an input to `buf` that performs the desired task.

- Your input can only contain percent formatters.
- Each blank can only contain exactly one of these formatters: `c`, `hn`, `n`, `s`, or `x`.
- You **may not** pad the output of a formatter (e.g. `%5c` is not permitted).
- You can assume that writing to any address will not crash the program.
- Not all blanks need to be used, but you may not use more blanks than provided.

Note: `printf("%x", 0x1234001D)` will output `1234001d`. Notice how there is no prefixing `0x` and all letters are in lowercase.

Q6.1 (6 points) Print a string containing `evanbot`. (The output can have other characters too, but `evanbot` needs to be in the output.)

%__________ %__________ %__________ %__________ %__________ %__________

Q6.2 (6 points) Write the integer 13 to address `0xdeadbeef`.

%__________ %__________ %__________ %__________ %__________ %__________

Q6.3 (6 points) Print a string containing exactly the capital letter `A`, and nothing else.

Remember that the hexadecimal ASCII value for the character `A` is `0x41`.

%__________ %__________ %__________ %__________ %__________ %__________

Q6.4 (10 points) For this subpart, suppose that after `printf(buf)` returns, you overwrite `buf` with a second input, and then you immediately call `printf(buf)` again. The stack looks exactly the same on the second `printf` call, except any changes to memory from the first call will carry over to the second call.

You want the second `printf` call to output exactly `caketasty` and nothing else. It doesn't matter what the first `printf` call outputs.

Hint: What is the address of the string `caketasty` in memory?

Hint: Note that `0x20` is 32 in decimal.

First input to `buf`:

%__________ %__________ %__________ %__________ %__________ %__________

Second input to `buf`:

%__________ %__________ %__________ %__________ %__________ %__________

Q7 *Memory Safety Exploit: Valentine's Day* (22 points)

EvanBot wrote a program to exchange valentines with CodaBot. Mallory got her hands on EvanBot's code, and wants to send a valentine that executes her shellcode and ruins Valentine's day!

```
1  typedef struct message {
2      char *ptr;
3      char text[64];
4  } message_t;
5
6  typedef struct reply {
7      char *ptr;
8      char text[8];
9  } reply_t;
10
11 void valentine() {
12     reply_t coda;
13     coda.ptr = &(coda.text[____]);
14     fgets(coda.ptr, 5, stdin);
15 }
16
17 void main(){
18     message_t evan;
19     reply_t bot;
20     fgets(evan.text, 64, stdin);
21     evan.ptr = &evan.text[0];
22     valentine();
23 }
```

Stack at Line 13:

RIP of `main`
SFP of `main`
canary
(1)
(2)
(3)
`bot.ptr`
RIP of `valentine`
SFP of `valentine`
(4)
(5)
(6)

ASLR is enabled on the stack, but not in the code section. (In other words, the stack section is moved on each run, but the code section stays in the same place.) **Stack canaries** are also enabled. No other memory safety defenses are enabled.

Q7.1 (2 points) What values go in Blanks (1)-(3) in the stack diagram above?

- ◯ (1) - `bot.text[8]` (2) - `evan.text[64]` (3) - `evan.ptr`
- ◯ (1) - `bot.text[8]` (2) - `evan.ptr` (3) - `evan.text[64]`
- ◯ (1) - `evan.text[64]` (2) - `evan.ptr` (3) - `bot.text[8]`
- ◯ (1) - `evan.ptr` (2) - `evan.text[64]` (3) - `bot.text[8]`

Q7.2 (2 points) What values go in Blanks (4)-(6) in the stack diagram above?

- ◯ (4) - `coda.text[8]` (5) - `coda.ptr` (6) - canary
- ◯ (4) - `coda.text[8]` (5) - canary (6) - `coda.ptr`
- ◯ (4) - canary (5) - `coda.text[8]` (6) - `coda.ptr`
- ◯ (4) - canary (5) - `coda.ptr` (6) - `coda.text[8]`

Mallory runs GDB and finds some useful assembly in the executable. (Assume that all x86 instructions are 4 bytes long.)

```
0x08988154: xor %ebx, %ebx
0x0898815c: mov %ebp, %esp
0x08988160: pop %edx
0x08988164: pop %ecx
0x08988168: pop %ebx
0x0898816c: pop %eax
0x08988170: pop %ebp
0x0898816c: ret
```

Create an exploit that would cause this program to execute shellcode.

Hints (there's no new information here, but it might give you ideas if you're stuck):

- The `pop` instruction takes the lowest value (where the stack pointer register `esp` is pointing) off the stack, and puts that value in the specified register. For example, `pop %eax` puts the lowest value on the stack into register `%eax`. When a value is popped off the stack, the `esp` moves up by 4.
- The `ret` instruction behaves like `pop %eip`.

Q7.3 (5 points) What integer should go in the blank on Line 13?

Q7.4 (9 points) What should be the inputs to the two `fgets` calls?

You can use `SHELLCODE` to represent the 60-byte shellcode you want to execute.

Input to `fgets` in `main`, at **Line 20**:

Input to `fgets` in `valentine`, at **Line 14**:

Q7.5 (4 points) Mallory runs GDB and discovers that the address of `bot.text` is `0xffff1234`.

Can Mallory use this address to exploit this program?

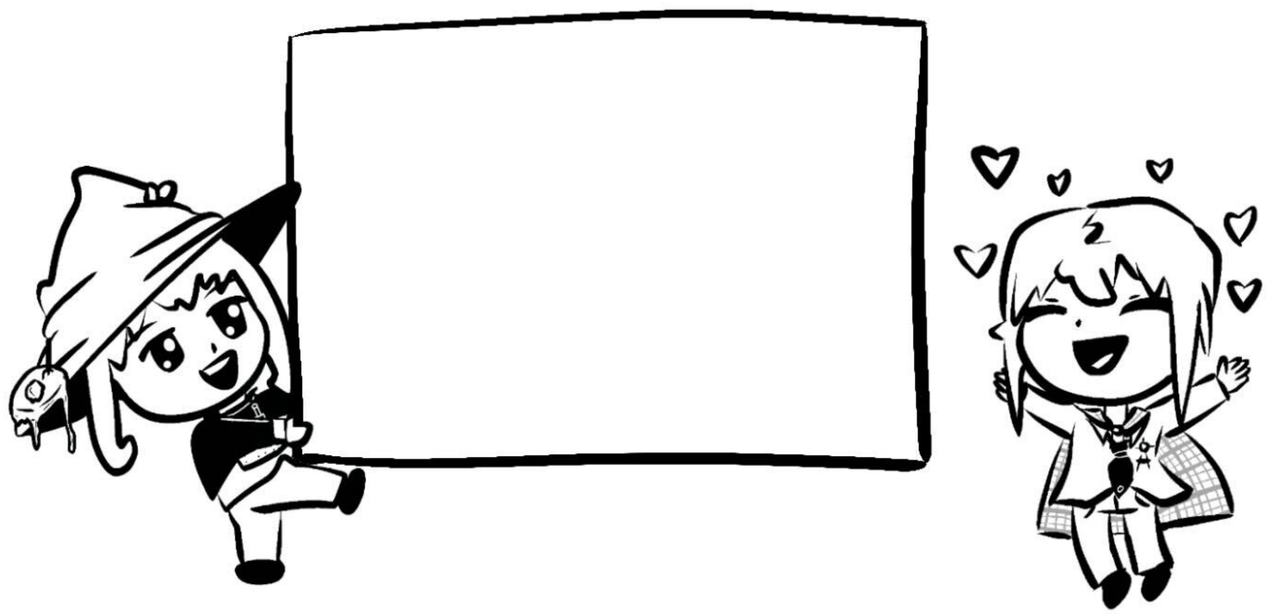
○ Yes ○ No

Briefly justify your answer. You can answer in 15 words or fewer.

Nothing on this page will affect your grade in any way.

Post-Exam Activity: Valentine

What was in EvanBot's valentine to CodaBot? Draw it here!



Comment Box

Congratulations for making it to the end of the exam! Feel free to leave any final thoughts, comments, feedback, or doodles here: