Q1  C Memory Defenses  
Mark the following statements as True or False and justify your solution. Please feel free to discuss with students around you.

1. Stack canaries completely prevent a buffer overflow from overwriting the return instruction pointer.

2. A format-string vulnerability can allow an attacker to overwrite values below the stack pointer.

3. An attacker exploits a buffer overflow to redirect program execution to their input. This attack no longer works if the data execution prevention/executable space protection/NX bit is set.

4. If you have a non-executable stack and heap, buffer overflows are no longer exploitable.

5. If you use a memory-safe language, some buffer overflow attacks are still possible.

6. ASLR, stack canaries, and NX bits all combined are insufficient to prevent exploitation of all buffer overflow attacks.
Short answer!

1. What vulnerability would arise if the canary was above the return address?

2. What vulnerability would arise if the stack canary was between the return address and the saved frame pointer?

3. Assume ASLR is enabled. What vulnerability would arise if the instruction \texttt{jmp *esp} exists in memory?
Q2  Robin  (20 points)

Consider the following code snippet:

```c
void robin(void) {
    char buf[16];
    int i;
    if (fread(&i, sizeof(int), 1, stdin) != 1)
        return;
    if (fgets(buf, sizeof(buf), stdin) == NULL)
        return;
    _________
}
```

Assume that:

- There is no compiler padding or additional saved registers.
- The provided line of code in each subpart compiles and runs.
- buf is located at memory address 0xffffd8d8
- Stack canaries are enabled, and all other memory safety defenses are disabled.
- The stack canary is four completely random bytes (no null byte).

For each subpart, mark whether it is possible to leak the value of the stack canary. If you put possible, provide an input to Line 5 and an input to Line 8 that would leak the canary. If the line is not needed for the exploit, you must write "Not needed" in the box.

Write your answer in Python 2 syntax (just like in Project 1).

Q2.1  (3 points) Line 11 contains `gets(buf);`.

- Possible
- Not possible

Line 5:

```
```

Line 8:

```
```
Q2.2 (5 points) For this subpart only, enter an input that allows you to leak a single character from memory address 0xffffd8d7. Mark “Not possible” if this is not possible. Line 11 contains `printf("%c", buf[i]);`.

- Possible
- Not possible

Line 5:

Line 8:

Q2.3 (6 points) Line 11 contains `printf(buf);`.

- Possible
- Not possible

Line 5:

Line 8:
Q2.4 (6 points) Line 11 contains `printf(i);`.

- Possible
- Not possible

Line 5:

Line 8:
Q3  IND-CPA  (0 points)

When formalizing the notion of confidentiality, as provided by a proposed encryption scheme, we introduce the concept of indistinguishability under a chosen plaintext attack, or IND-CPA security. A scheme is considered IND-CPA secure if an attacker cannot gain any information about a message given its ciphertext. This definition can be defined as an experiment between a challenger and adversary, detailed in the diagram below:

Consider the one-time pad encryption scheme discussed in class. For parts (a) - (c), we will prove why one-time pad is not IND-CPA secure and, thus, why a key should not be reused for one-time pad encryption.

Q3.1 With what messages $M_1$ and $M_0$ should the adversary provide the challenger?

Q3.2 Now, for which message(s) should the adversary request an encryption from the challenger during the query phase?

Q3.3 The challenger will now flip a random bit $b \in \{0, 1\}$, encrypt $M_b$, and send back $C = Enc(k, M_b) = M_b \oplus k$ to the adversary. How does the adversary determine $b$ with probability $\frac{1}{2}$?
Q3.4 Putting it all together, explain how an adversary can always win the IND-CPA game with probability 1 against a deterministic encryption algorithm. Note: Given an identical plaintext, a deterministic encryption algorithm will produce identical ciphertext.

Q3.5 Assume that an adversary chooses an algorithm and runs the IND-CPA game a large number of times, winning with probability 0.6. Is the encryption scheme IND-CPA secure? Why or why not?

Q3.6 Now, assume that an adversary chooses an algorithm and runs the IND-CPA game a large number of times, winning with probability 0.5. Is the encryption scheme IND-CPA secure? Why or why not?