Student ID: _____

This sheet will not be graded (feel free to write on it), but you must turn it in at the end of the exam.

C Function Definitions

```
int printf(const char *format, ...);

      printf() produces output according to the format string format.

      Conversion specifiers:
      %hn The number of characters printed so far is stored into the 2-byte
          short integer pointed to by the corresponding argument.
          Outputs no bytes.
      %n  The number of characters printed so far is stored into the integer
          pointed to by the corresponding argument.
          Outputs no bytes.
      %s  String (pointer to a character array).
          Outputs bytes until null terminator.
      %c  Character. Outputs 1 byte.
      %x  Hexadecimal. Outputs 8 bytes.

      Each of the above conversion specifiers reads a 4-byte argument on the stack.

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.
```

General Exam Assumptions

Unless otherwise specified, you can assume these facts on the entire exam:

- Memory safety:
  - You are on a little-endian 32-bit x86 system.
  - There is no compiler padding or saved additional registers.
  - If stack canaries are enabled, they are four completely random bytes (no null byte).
  - You can write your answers in Python syntax (as seen in Project 1).
- Cryptography:
  - The attacker knows the algorithms being used (Shannon's maxim).
  - $\|$ denotes concatenation.
  - H refers to a secure cryptographic hash function.

Valentines' Day: Code

Below is the code and the stack diagram in the *Valentine's Day* question, repeated for your convenience.

```
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 }
```

This diagram is **ungraded**.

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

Cake Without Pan: Diagram

Below is the stack diagram in the *Cake Without Pan* question, repeated for your convenience.

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`