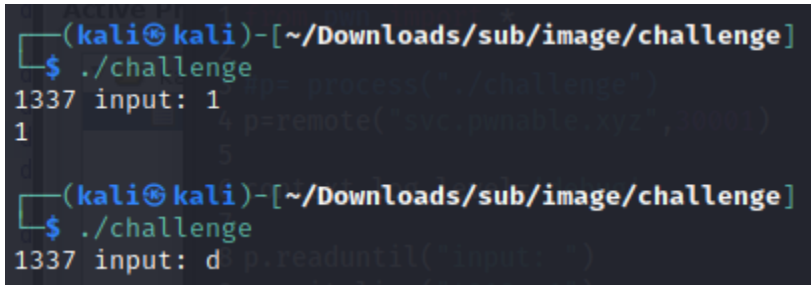


Practice #4

Challenge 2: Sub

- For this challenge I downloaded the binary and ran the program on the command line. I noticed that the program required two inputs, and that it did not allow any character inputs, or else it terminated immediately.



```
(kali㉿kali)-[~/Downloads/sub/image/challenge]
$ ./challenge
1337 input: 1
1
(kali㉿kali)-[~/Downloads/sub/image/challenge]
$ ./challenge
1337 input: d
```

- I then passed the binary into Ghidra to see what the decompiler would give me, and then I noticed this piece of code:

```
__isoc99_scanf("%u %u", &local_18, &local_14);
if ((local_18 < 0x1337) && (local_14 < 0x1337)) {
    if (local_18 - local_14 == 0x1337) {
        system("cat /flag");
    }
}
```

- This tells me that there are two variables where both must be less than the hex value of 0x1337 (which is 4919 in decimal), and that the difference between both variables must also be 4919, and if that condition is satisfied would a flag file be printed to the console. The scanf also tells me that the second input is being subtracted from the first input. It is mathematically impossible that the variables local_18 and local_14 are both positive, so I looked at the types of the variables towards the top of the main function and noticed this:

```
long in_FS_OFFSET;
int local_18;
int local_14;
long local_10;
```

- This tells me that the variables that we are working with are signed integers, which means that they can be negative. If I pass in the first argument as 4918 and the second argument as -1 the

terminal would return as such:

```
(kali㉿kali)-[~/Downloads/sub/image/challenge]
$ ./challenge
1337 input: 4918 -1
cat: /flag: No such file or directory
```

- The reason why the terminal says /flag does not exist is because we're running the program locally, and if we want the flag we would have to pass that input into the server. The python code would look something like this:

```
1 from pwn import *
2
3 #p= process("./challenge")
4 p=remote("svc.pwnable.xyz",30001)
5
6 context.log_level='debug'
7
8 p.readuntil("input: ")
9 p.writeline("4918 -1")
10 p.interactive()
```

- Since the server was down at the time I made this write up, I was unable to retrieve the flag. However, since the local program was able to get to the `cat /flag` part of the program that would be the correct input to the server.