HOMEWORK 5

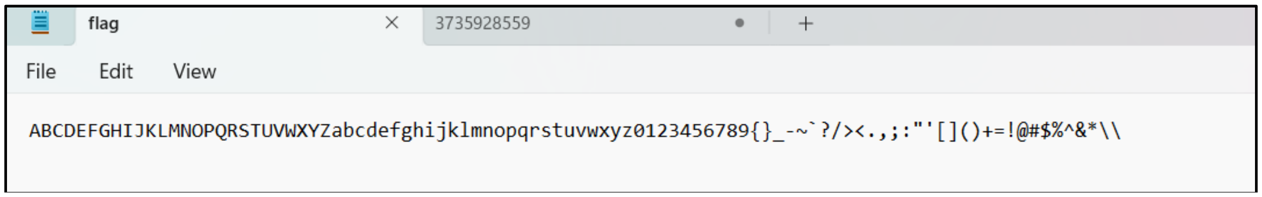
~Purva Naresh Rumde (pr23b)

Problem ID	Captured Flag	Steps
P1	FSUctf{fr_th0_th4t_w45_n0_j0k3}	In this problem, we received a Python encryption program for flag.txt. We provided input characters including uppercase letters, lowercase letters, numbers, and some special characters. The program generated a sequence of numbers as output. We mapped these numbers to our input characters by comparing them with the provided input. By matching and mapping the characters, we decoded the flag.
P2	fsuctf{b1t5_0per4t0r5_4re_c00l}	In this problem, we received an encryptor file and analyzed it using IDA. We discovered a shiftCharacters function that shifted characters 5 times. Using Python, we converted the provided output to binary, applied a shift cipher, and converted it back to ASCII. This process yielded the flag.
P3	3735927357	In this problem, we analyzed the program in IDA and found a reference to the value 0xDEADBEEF, which appeared to be related to a key. We also identified an encrypt function containing the hex value 0x42D. By XORing these values, we obtained the hex value 0xDEADBA3D, which we converted to decimal to derive the flag.
P4	flag{M00_R06_P06}	In this problem, we provided inputs for the number of classes missed, our level of knowledge in software reverse engineering, and our proficiency in binary exploitation. With the inputs of 0, 6, and 6 respectively, we obtained the flag.

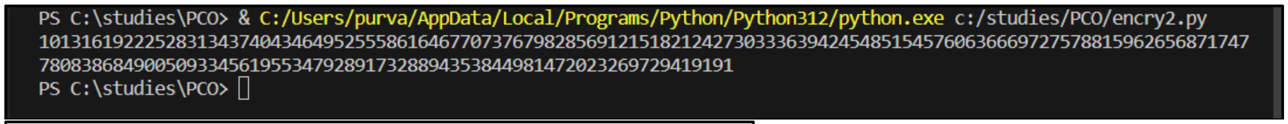
Detailed Explanations (Including Screenshots)

Q1.

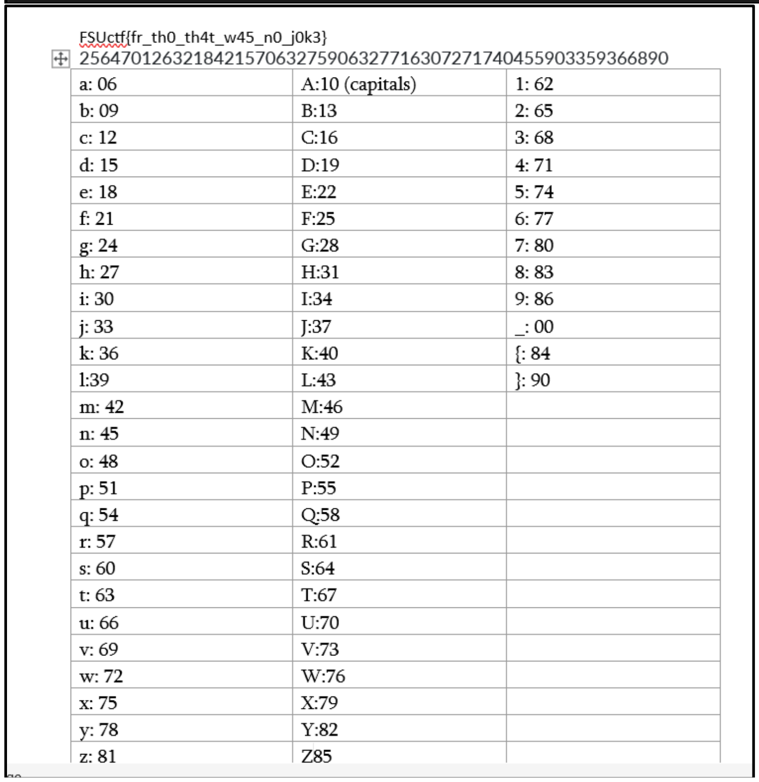
- In this problem we had been provided with a python program that does encryption for the flag.txt.
- So I provided the alphabets from in capital, small, numbers and a few special characters.
- I got an output of numbers that I had to map with the input that I had given.
- I then compared and mapped the characters with the input that was provided along with the code file and that's how I got the flag.



```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789{}_-~`?/><.,;:"'[]()+=!@#$%^&*\\
```



```
PS C:\studies\PCO> & C:/Users/purva/AppData/Local/Programs/Python/Python312/python.exe c:/studies/PCO/encry2.py
1013161922252831343740434649525558616467707376798285691215182124273033363942454851545760636669727578815962656871747
7808386849005093345619553479289173288943538449814720232697294191
PS C:\studies\PCO> 
```

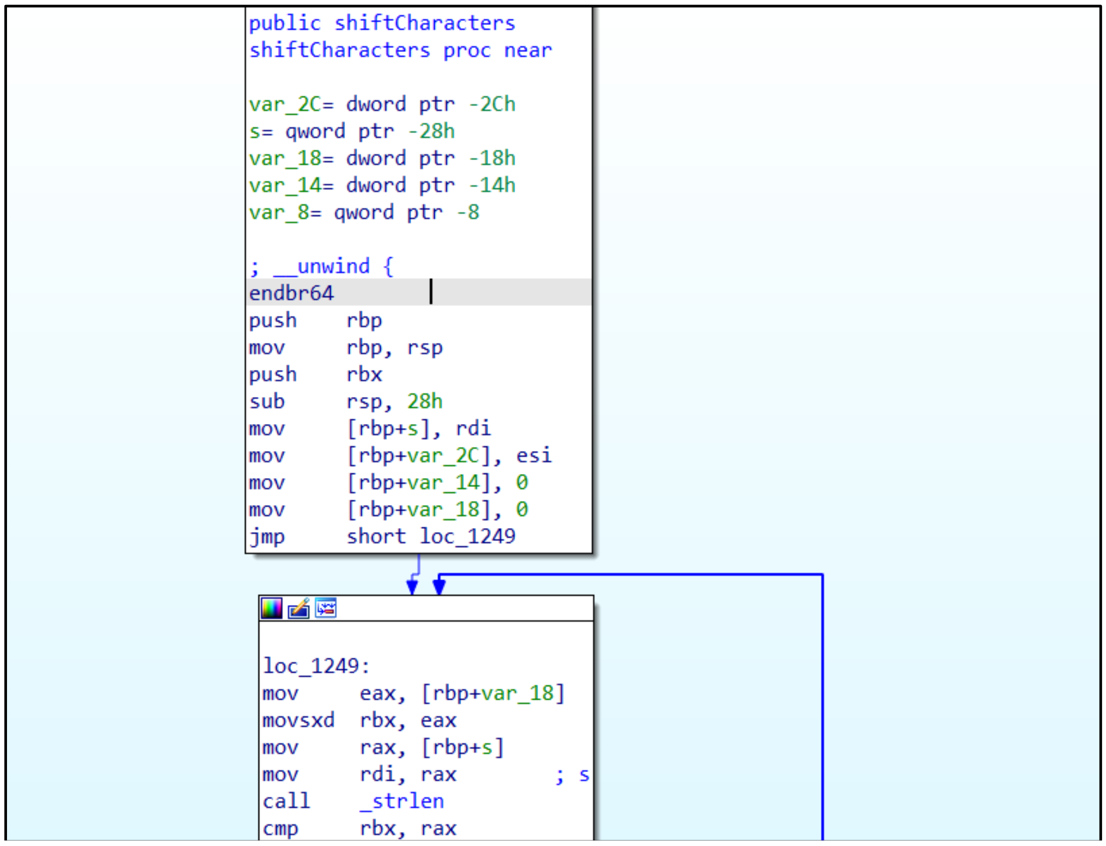


FSUctf{fr_th0_th4t_w45_n0_j0k3}
25647012632184215706327590632771630727174045590335936 6890

a: 06	A:10 (capitals)	1: 62
b: 09	B:13	2: 65
c: 12	C:16	3: 68
d: 15	D:19	4: 71
e: 18	E:22	5: 74
f: 21	F:25	6: 77
g: 24	G:28	7: 80
h: 27	H:31	8: 83
i: 30	I:34	9: 86
j: 33	J:37	_: 00
k: 36	K:40	{: 84
l:39	L:43	}: 90
m: 42	M:46	
n: 45	N:49	
o: 48	O:52	
p: 51	P:55	
q: 54	Q:58	
r: 57	R:61	
s: 60	S:64	
t: 63	T:67	
u: 66	U:70	
v: 69	V:73	
w: 72	W:76	
x: 75	X:79	
y: 78	Y:82	
z: 81	Z85	

Q2.

- In this problem we had been provided with encryptor file.
- That I opened in IDA.
- After studying the file I got to know that it consists of a shiftCharacters function that shifts characters 5 times.
- I wrote a program in python that took the output provided with the question as an input and converted it to binary and then used shift cipher on it and then again converted it to ASCI.
- And then I got the flag.

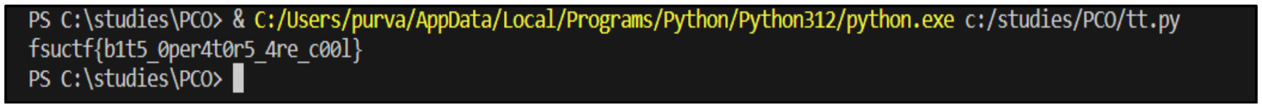


```
public shiftCharacters
shiftCharacters proc near

var_2C= dword ptr -2Ch
s= qword ptr -28h
var_18= dword ptr -18h
var_14= dword ptr -14h
var_8= qword ptr -8

; __unwind {
endbr64
push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 28h
mov     [rbp+s], rdi
mov     [rbp+var_2C], esi
mov     [rbp+var_14], 0
mov     [rbp+var_18], 0
jmp     short loc_1249
```

```
loc_1249:
mov     eax, [rbp+var_18]
movsxd  rbx, eax
mov     rax, [rbp+s]
mov     rdi, rax        ; s
call    _strlen
cmp     rbx, rax
```

```
# Function to convert decimal to binary
def decimal_to_binary(n):
    return bin(n)[2:]

# Function to convert binary to ASCII
def binary_to_ascii(b):
    return ''.join(chr(int(b[i:i+8], 2)) for i in range(0, len(b), 8))

# Decimal numbers
```

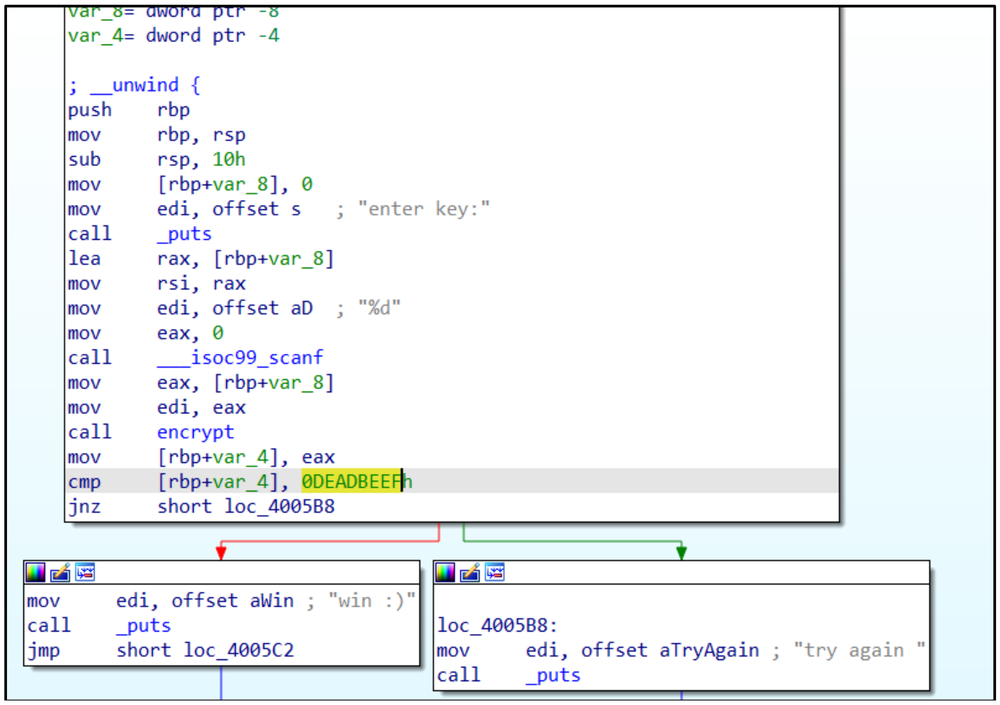
```
decimal_numbers = [3264, 3680, 3744, 3168, 3712, 3264, 3936, 3136, 1568, 3712, 1696, 3040,
1536, 3584, 3232, 3648, 1664, 3712, 1536, 3648, 1696, 3040, 1664, 3648, 3232, 3040, 3168,
1536, 1536, 3456, 4000]

# Convert to binary, shift, and then convert back to ASCII
for num in decimal_numbers:
    binary_rep = decimal_to_binary(num)
    shifted_binary_rep = binary_rep[:-5]  # Shift 5 times to the right
    ascii_text = binary_to_ascii(shifted_binary_rep)
    print(ascii_text, end='')
```



```
PS C:\studies\PCO> & C:/Users/purva/AppData/Local/Programs/Python/Python312/python.exe c:/studies/PCO/tt.py
fsuctf{b1t5_0per4t0r5_4re_c00l}
PS C:\studies\PCO> 
```

Q3.

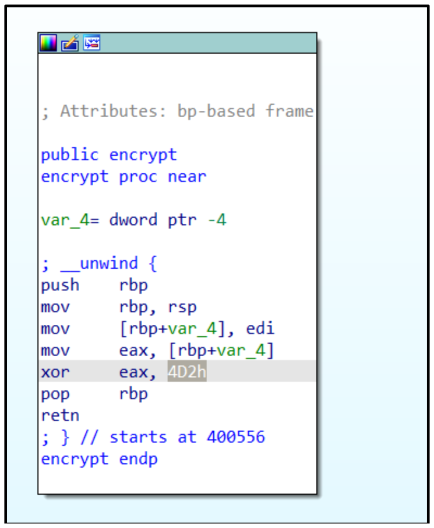
- In this problem similarly I opened IDA and studied the whole program and stumbled upon the 0DEADBEEFh.
- It somehow relates with the key and compares the key to match it.
- There is also a function called encrypt that consists of this hex value 42Dh.
- After XORing these 2 values I got a hex value deadba3d.
- I converted this value to decimal and got the flag.



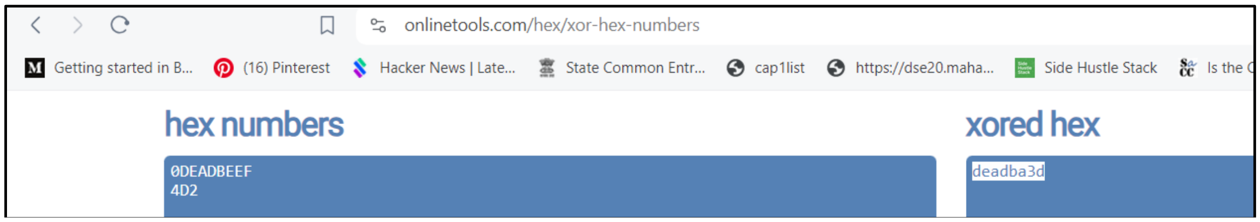
```
var_8= dword ptr -8
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 10h
mov     [rbp+var_8], 0
mov     edi, offset s   ; "enter key:"
call    _puts
lea     rax, [rbp+var_8]
mov     rsi, rax
mov     edi, offset aD  ; "%d"
mov     eax, 0
call    ___isoc99_scanf
mov     eax, [rbp+var_8]
mov     edi, eax
call    encrypt
mov     [rbp+var_4], eax
cmp     [rbp+var_4], 0DEADBEEFh
jnz     short loc_4005B8
```

```
mov     edi, offset aWin ; "win :)"
call    _puts
jmp     short loc_4005C2
```

```
loc_4005B8:
mov     edi, offset aTryAgain ; "try again "
call    _puts
```



```
; Attributes: bp-based frame

public encrypt
encrypt proc near

var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
mov     [rbp+var_4], edi
mov     eax, [rbp+var_4]
xor     eax, 4D2h
pop     rbp
retn
; } // starts at 400556
encrypt endp
```



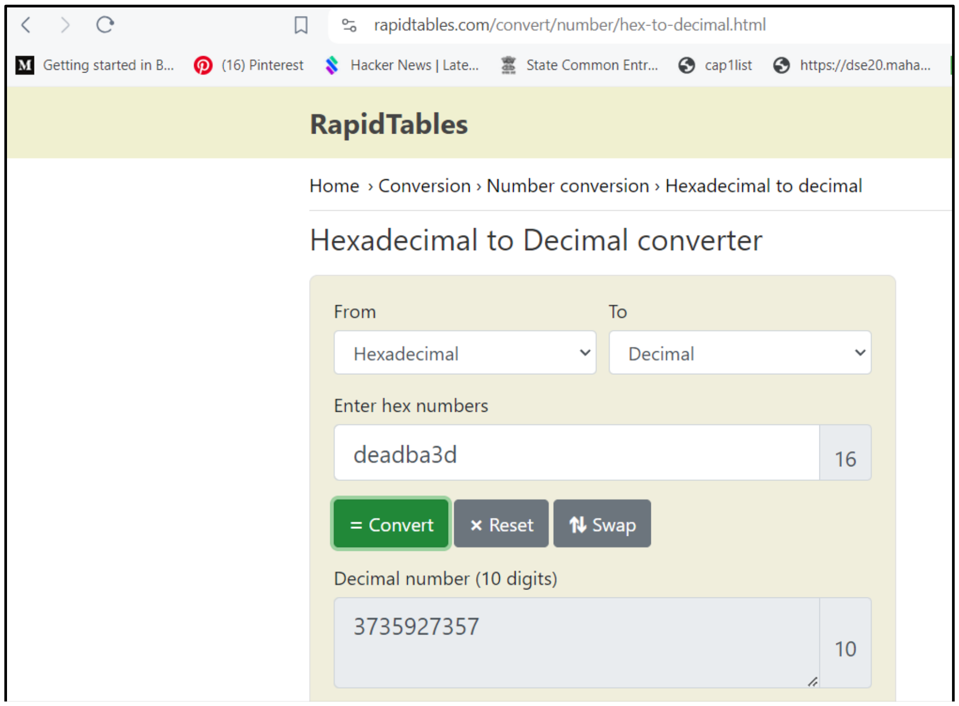
onlinetools.com/hex/xor-hex-numbers

hex numbers

```
0DEADBEEF
4D2
```

xored hex

```
deadba3d
```



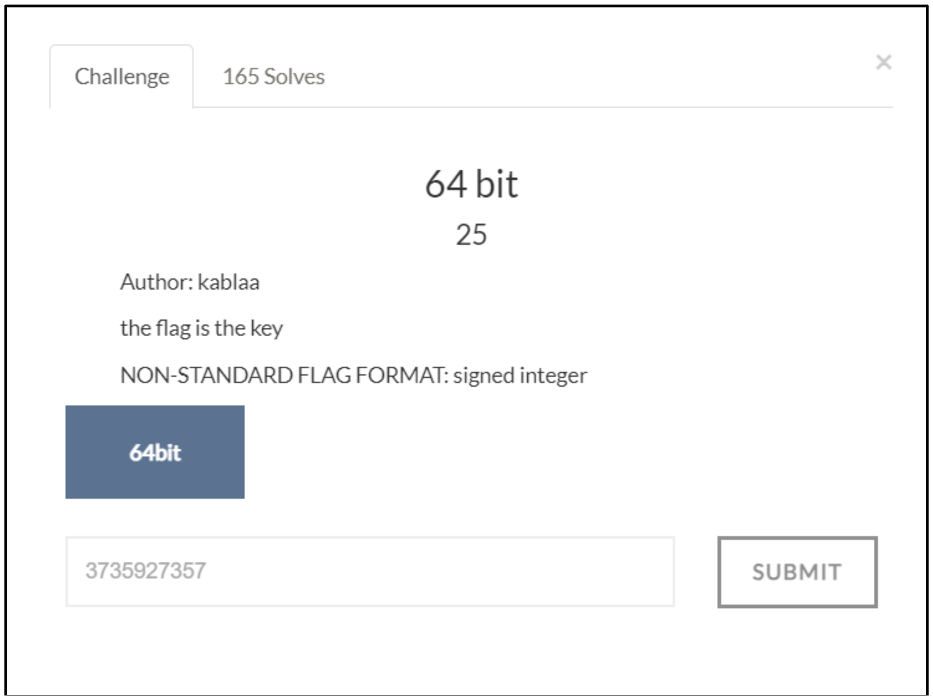
rapidtables.com/convert/number/hex-to-decimal.html

Hexadecimal to Decimal converter

From: Hexadecimal
To: Decimal

Enter hex numbers: deadba3d (16)

Decimal number (10 digits): 3735927357 (10)



64 bit

25

Author: kablaa

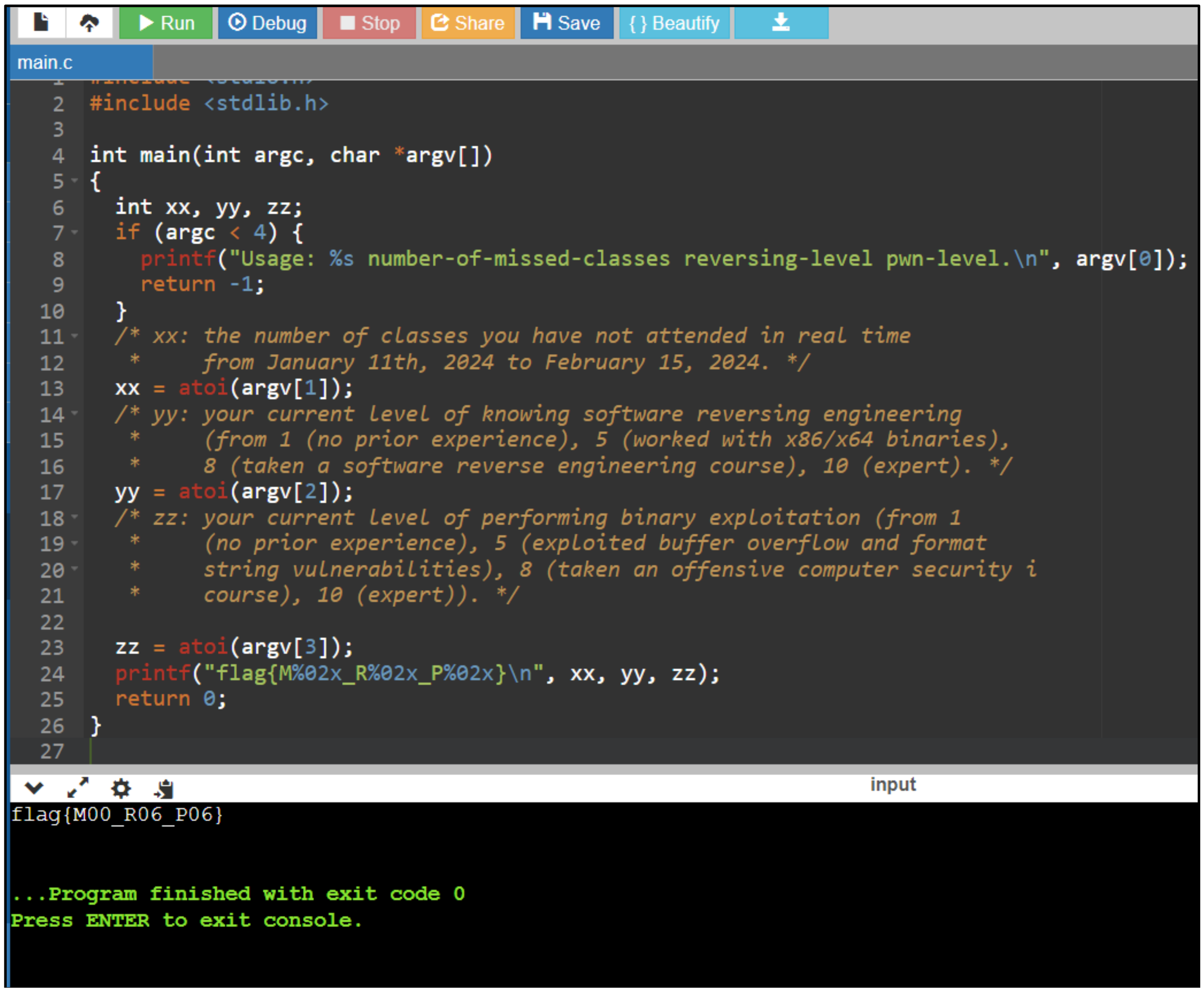
the flag is the key

NON-STANDARD FLAG FORMAT: signed integer

3735927357

Q4.

- In this problem we just had to give the inputs that were asked.
- First one for how many number of classes not attended during a given period for which I had entered 0
- Second my current level of knowing software reversing engineering which I answered as 6
- Third my current level of performing binary exploitation which I answered as 6.
- And that's how I got the flag.



```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  int xx, yy, zz;
  if (argc < 4) {
    printf("Usage: %s number-of-missed-classes reversing-level pwn-level.\n", argv[0]);
    return -1;
  }
  /* xx: the number of classes you have not attended in real time
   *     from January 11th, 2024 to February 15, 2024. */
  xx = atoi(argv[1]);
  /* yy: your current level of knowing software reversing engineering
   *     (from 1 (no prior experience), 5 (worked with x86/x64 binaries),
   *     8 (taken a software reverse engineering course), 10 (expert). */
  yy = atoi(argv[2]);
  /* zz: your current level of performing binary exploitation (from 1
   *     (no prior experience), 5 (exploited buffer overflow and format
   *     string vulnerabilities), 8 (taken an offensive computer security i
   *     course), 10 (expert)). */

  zz = atoi(argv[3]);
  printf("flag{M%02x_R%02x_P%02x}\n", xx, yy, zz);
  return 0;
}
```

```
flag{M00_R06_P06}


...Program finished with exit code 0
Press ENTER to exit console.
```