why

使用ghidra反編譯，首先看到main:

```
undefined8 main(void)

{
  int local_c;

  printf("Give me flag: ");
  __isoc99_scanf(&DAT_00102027,buf);
  local_c = 0;
  while( true ) {
    if (0x18 < local_c) {
      pass = 1;
      return 0;
    }
    if ((byte)buf[local_c] - 10 != (uint)(byte)(&enc_flag)[local_c]) break;
    local_c = local_c + 1;
  }
  return 0;
}

```

solve(failed)

這個main的邏輯非常簡單，直接寫程式嘗試解密flag:

```
data = [0x50, 0x56, 0x4b, 0x51, 0x85, 0x73, 0x78, 0x73, 0x7e, 0x69, 0x70,
 ↪  0x73,
        0x78, 0x73, 0x69, 0x77, 0x7a, 0x7c, 0x79, 0x7e, 0x6f, 0x6d, 0x7e,
         ↪  0x2b, 0x87]
flag = [chr(i+10) for i in data]
print("".join(flag))
```

結果程式跑出一堆亂碼，看來程式其他部分有做了什麼事情，讓這個題目不如看起來的簡單。

init / fini

init與fini是程式的兩個特別的部分，init會在entry之前被執行，而fini則是在程式離開前最後會被執行。來看看這個程式的init與fini分別做了什麼:

```
                        __DT_INIT_ARRAY                                 XREF[6]
                        __init_array_start
                        __frame_dummy_init_array_entry

    00103db0 60 11 10      dq        frame_dummy
             00 00 00
             00 00

                        QWORD_00103db8                                  XREF[1]
    00103db8 69 11 10      dq        sub_1169
             00 00 00
             00 00
    00103dc0 9d 11 10      dq        sub_119d
             00 00 00
             00 00
    00103dc8 d8 11 10      dq        sub_10d8
             00 00 00
             00 00
```

```
                    __DT_FINI_ARRAY                                 XREF[3]
                    __init_array_end
                    __do_global_dtors_aux_fini_array_entry
  00103dd0 20 11 10      dq        __do_global_dtors_aux
           00 00 00
           00 00
  00103dd8 f8 11 10      dq        sub_11f8
           00 00 00
           00 00
```

init

```
void sub_1169(void)

{
  main_page = _init;
  mprotect(_init,0x1000,7);
  return;
}
```

把main_page設成_init的位置，其中_init的位置是0x101000。使用mprotect把_init後0x1000的記憶體位置改

```
void sub_119d(void)

{
  byte *pbVar1;

  pbVar1 = (byte *)(main_page + 0x283);
  *pbVar1 = ~*pbVar1;
  *pbVar1 = *pbVar1 + 1;
  return;
}
```

把main_page+0x283的位置做neg(二補數)運算

```
void sub_10d8(void)

{
  mprotect(main_page,0x1000,5);
  return;
}
```

使用mprotect把_init後0x1000的記憶體位置改回可執行

fini

```
1
2  void sub_11f8(void)
3
4  {
5    if (pass == '\0') {
6      puts("Wrong :(");
7    }
8    else {
9      puts("Correct :)");
10   }
11   return;
12 }
13
```

這邊就是沒有在main看到的output

init改了0x101000 + 0x283位置的記憶體，所以看到0x101283是什麼東西:

```
0010127a 0f b6 04 10     MOVZX     EAX,byte ptr [RAX + RDX*0x1]=>buf
0010127e 0f b6 c0        MOVZX     EAX,AL
00101281 8d 48 f6        LEA       ECX,[RAX + -0xa]
00101284 8b 45 fc        MOV       EAX,dword ptr [RBP + local_c]
00101287 48 98           CDQE
00101289 48 8d 15 a0 2d  LEA       RDX,[enc_flag]
```

他把f6改掉了，改成f6的二補數，也就是會從-0xa變成0xa，這也是爲什麼一開始解題的程式碼是錯誤的，要改

solve

已經找到原因，稍微修改程式碼:

```
data = [0x50, 0x56, 0x4b, 0x51, 0x85, 0x73, 0x78, 0x73, 0x7e, 0x69, 0x70,
 ↪  0x73,
        0x78, 0x73, 0x69, 0x77, 0x7a, 0x7c, 0x79, 0x7e, 0x6f, 0x6d, 0x7e,
         ↪  0x2b, 0x87]
flag = [chr(i-10) for i in data]
print("".join(flag))
```

```
$ python3 sol.py
FLAG{init_fini_mprotect!}
```