Challenge 1:

Goal:
Make the executable print “You did it!”

Steps taken:
The first step was to immediately go into the Symbol Tree and jump to **main** In the right Decompile, I immediately spot:

47 | *if (local_9c == 2) {*
48 | *puts("You did it!");*

Additionally there are a few requirements, for example:

20 | *if (param_1 == 3) { // argc must be 3*
21 | *__stream = fopen("strings","r"); // “strings” file is needed*

Let’s investigate where and how argv [named iVarl] is used:

30 | *iVar3 = strncmp(local_94,*(char **)(iVar1 + 4),0x14);*
39 | *iVar3 = strncmp(local_94,*(char **)(iVar1 + 8),0xc);*

Finally, there are some custom functions *f1* and *f2*. However, they’re very simple.

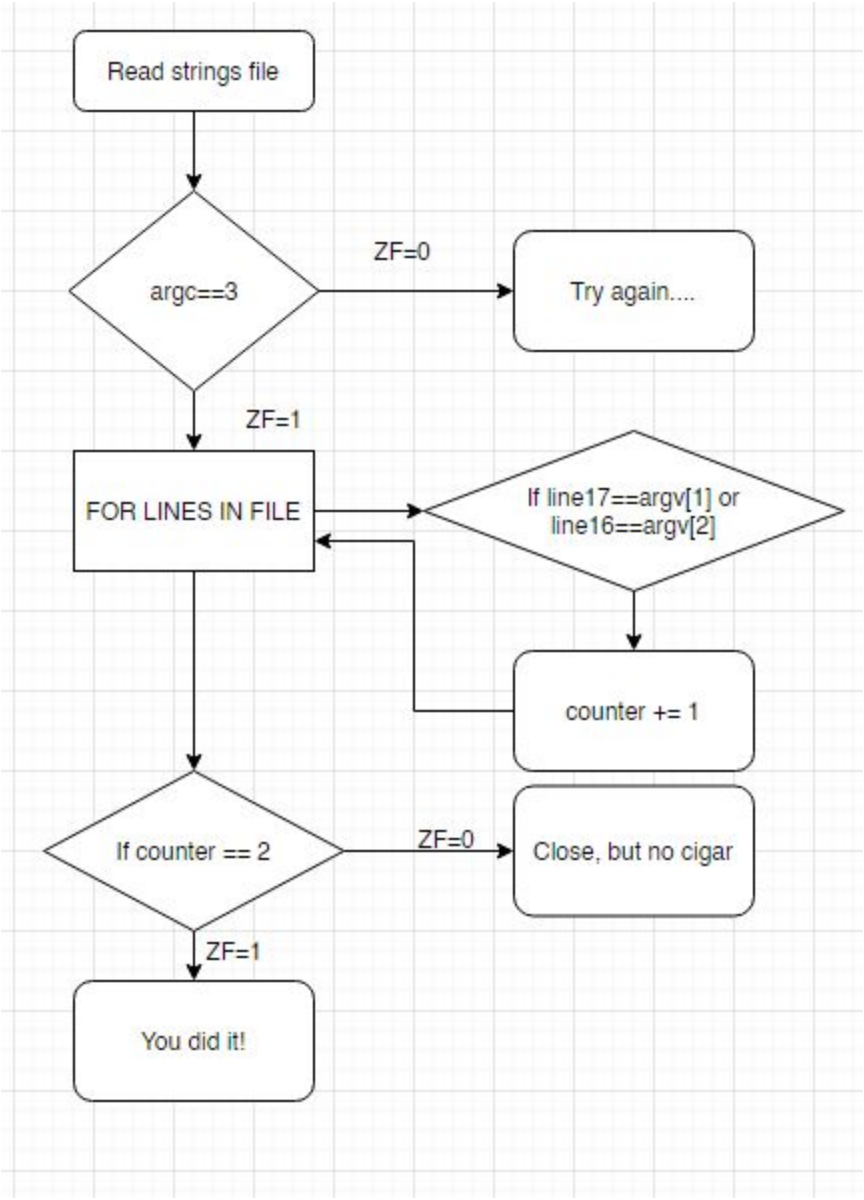
f1(x) = x+5
f2(x) = x+3

But considering they’re used on constants

27 | *iVar3 = f1(0xc); // iVar3 = 17*
36 | *iVar3 = f2(0xd); // iVar3 = 16*

So essentially the program is looking at lines 17 in the strings file for argv[1] and line 16 for argv[2].

Small CFD



Answer

kevin@ubuntu:~/Documents/c451/lab1$ ./re_challenege1 deregister_tm_clones __JCR_LIST__
__JCR_LIST__
deregister_tm_clones
You did it!

Challenge 2

Goal
print “You are great at this :)”

Steps taken:
Since the password is literally printed on the line before, getting it is quite simple. The actual password is very sneakily encoded though, a cast is performed on a large array of supposedly unused values to a char array.

There’s also some sneaky text such as:
could this be it?
M${aybe}__t{hi}s {onpizza_{time_}
hello world
you can do it
re_is_{ez}
QUFBQUFBQUFBQUFBQUFBQQ (base64 AAAAAA…)
RkxBR2ZsyagababaZ0ZMQUdmbGF

RkxBR2ZsYWdGTEFHZmxhZ0ZMQUdmbGF= (base64 FLAGflagFLAGflagFLAGfla)

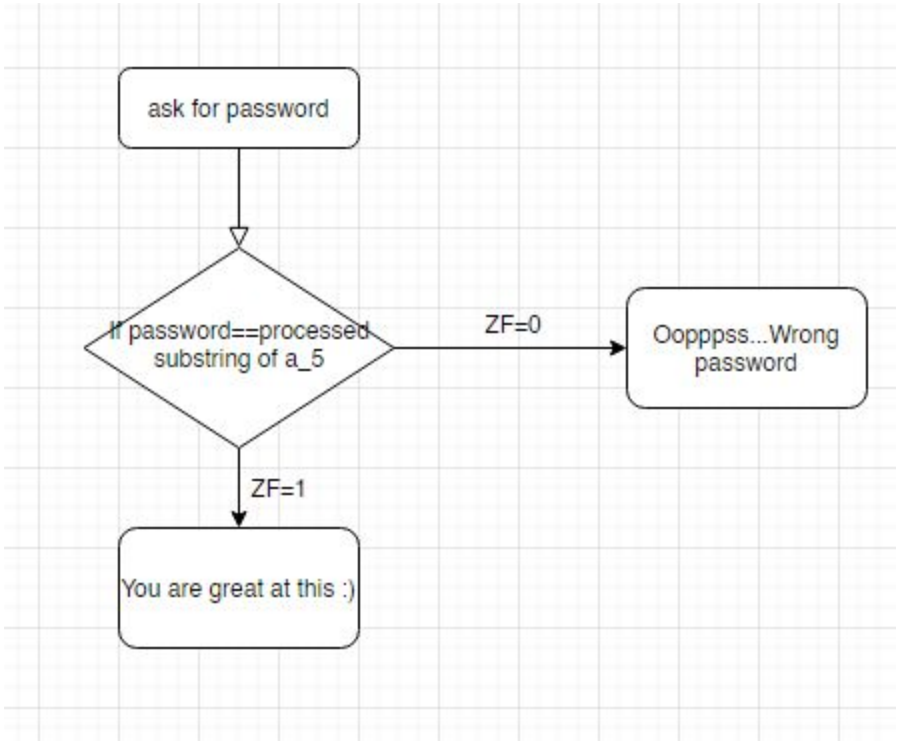
Q2FuIHlvdSByZWNvZ25p (base64 Can you recogni)

```
080486e1 c7 85 63 ff ff ff 52 6b 78 42    MOV    dword ptr [EBP + local_a5[0]],"RkxB"
080486eb c7 85 67 ff ff ff 52 32 5a 73    MOV    dword ptr [EBP + local_a5[4]],"R2Zs"
080486f5 c7 85 6b ff ff ff 79 61 67 61    MOV    dword ptr [EBP + local_a5[8]],"yaga"
080486ff c7 85 6f ff ff ff 62 61 62 61    MOV    dword ptr [EBP + local_a5[12]],"baba"
08048709 c7 85 73 ff ff ff 5a 30 5a 4d    MOV    dword ptr [EBP + local_a5[16]],"Z0ZM"
08048713 c7 85 77 ff ff ff 51 55 64 6d    MOV    dword ptr [EBP + local_a5[20]],"QUdm"
0804871d c7 85 7b ff ff ff 62 47 46 00    MOV    dword ptr [EBP + local_a5[24]],"bGF\x00"
```

```
28 | strtok(local_a5,"s");
29 | __s = strtok((char *)0x0,"\"");
30 | puts(__s);
31 | __s = strtok(__s,"Z");
```

So clearly, we just take everything after the lowercase “s” and before the upper “Z” which leaves a password of “yagababa”

Small CFD



Answer:

```
kevin@ubuntu:~/Documents/c451/lab1$ ./re_challenge2
Input your password...
yagababa
you entered yagababa
yagababaZ0ZMQUdmbGF
yagababa
You are great at this :)
```

Challenge 3

Goal:
Guess Flag

Steps taken:
Presumably, the answer lies in

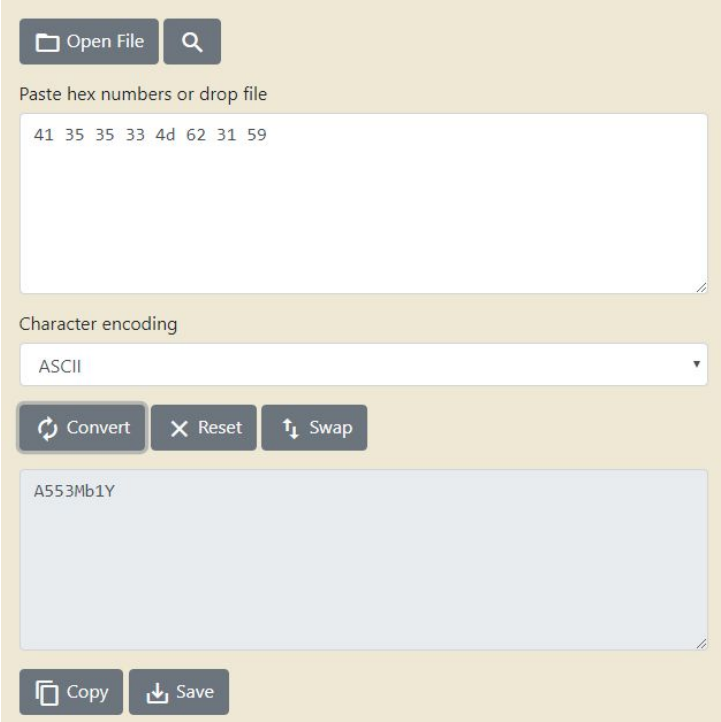
26 | *puts("You are amazing!!");*

In order to reach this code, you need esp-lc to equal the input in argv[1] (8 character's worth).

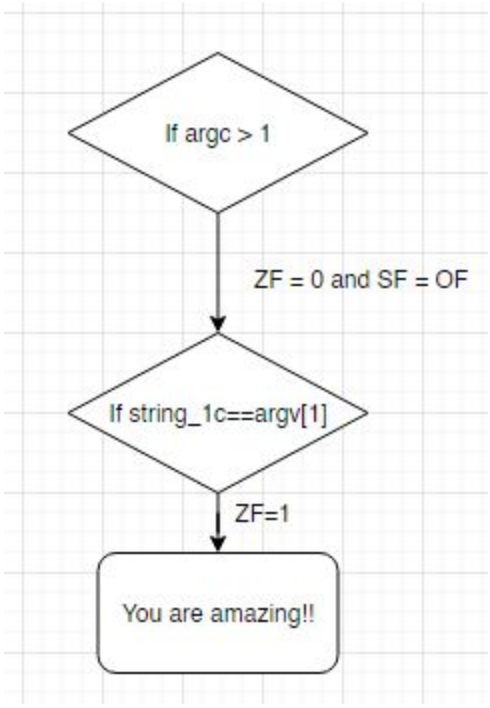
The same trick is used as last time. The string is encoded as a hex-array.

```
080485f1 c6 45 ec 41     MOV    byte ptr [EBP + local_1c],0x41
080485f5 c6 45 ed 35     MOV    byte ptr [EBP + local_1c+0x1],0x35
080485f9 c6 45 ee 35     MOV    byte ptr [EBP + local_1c+0x2],0x35
080485fd c6 45 ef 33     MOV    byte ptr [EBP + local_1c+0x3],0x33
08048601 c6 45 f0 4d     MOV    byte ptr [EBP + local_18],0x4d
08048605 c6 45 f1 62     MOV    byte ptr [EBP + local_18+0x1],0x62
08048609 c6 45 f2 31     MOV    byte ptr [EBP + local_18+0x2],0x31
0804860d c6 45 f3 59     MOV    byte ptr [EBP + local_18+0x3],0x59
08048611 c7 45 d8 00     MOV    dword ptr [EBP + local_30],0x0
```

Just convert it in ascending order.



Small CFD



Answer:

kevin@ubuntu:~/Documents/c451/lab1$ ./re_challenge3 A553Mb1Y
The answer: 1
Maybe it's this:5
You are amazing!!