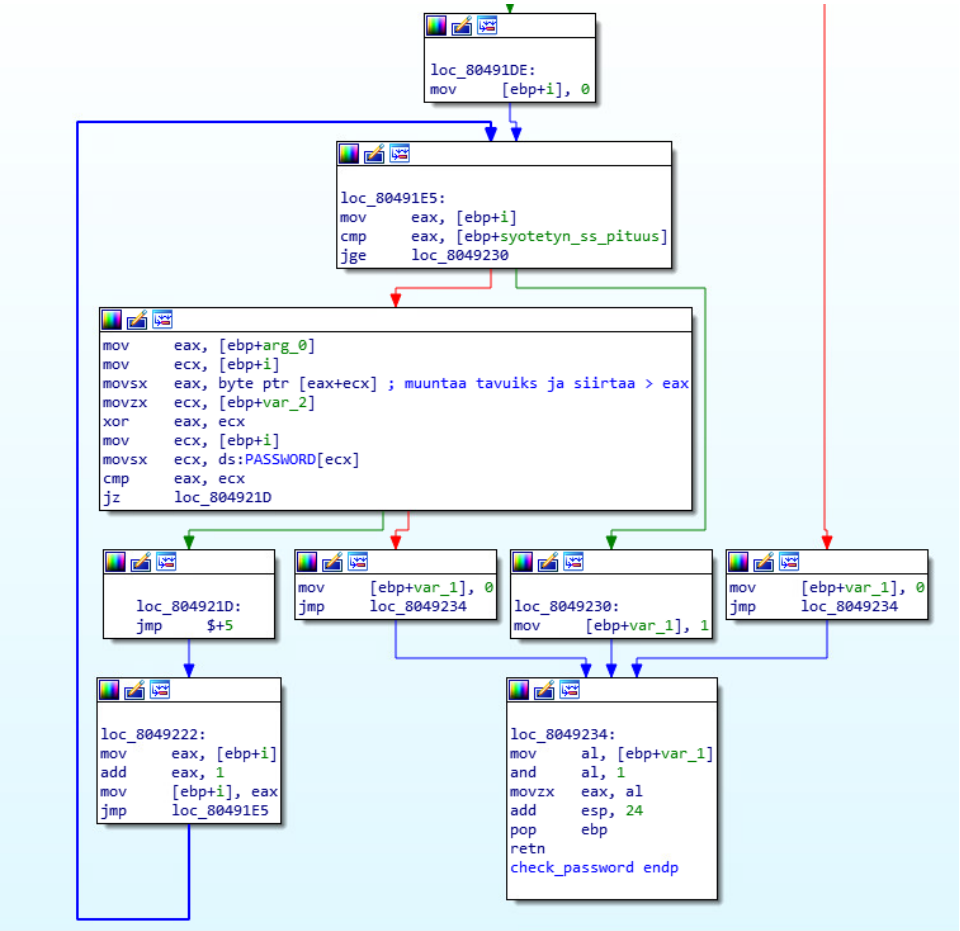
%s means that input is taken as string.

```
push    ebp
mov     ebp, esp
push    esi
sub     esp, 44h
mov     eax, [ebp+argv]
mov     ecx, [ebp+argc]
xor     edx, edx
mov     [ebp+var_8], 0
lea     esi, [ebp+var_26]
mov     [esp], esi
mov     dword ptr [esp+4], 0
mov     dword ptr [esp+8], 1Eh
mov     [ebp+var_2C], eax
mov     [ebp+var_30], ecx
mov     [ebp+var_34], edx
call    _memset
lea     eax, aPassword  ; "Password: "
mov     [esp], eax
call    _printf
lea     ecx, [ebp+var_26]
lea     edx, aS         ; "%s"
mov     [esp], edx
mov     [esp+4], ecx
mov     [ebp+var_38], eax
call    ___isoc99_scanf
lea     ecx, [ebp+var_26]
mov     [esp], ecx
mov     [ebp+var_3C], eax
call    check_password
and     al, 1
movzx   ecx, al
cmp     ecx, 1
jnz     loc_80492DF
```

First _strlen takes the length of right password and stores it to eax. Then moves it to variable (syotetyn_ss_pituus) Second _strlen takes inputs length and stores it to eax. Then it compares these two and if they match it goes to loop.

```
push    ebp
mov     ebp, esp
sub     esp, 18h
mov     eax, [ebp+arg_0]
mov     [ebp+var_2], 65
mov     ecx, esp
mov     dword ptr [ecx], offset PASSWORD
mov     [ebp+var_10], eax
call    _strlen
mov     [ebp+syotetyn_ss_pituus], eax
mov     eax, [ebp+arg_0]
mov     ecx, esp
mov     [ecx], eax
call    _strlen
cmp     eax, [ebp+syotetyn_ss_pituus] ; vertaa kahta strlen tulosta
jz      loc_80491DE
```

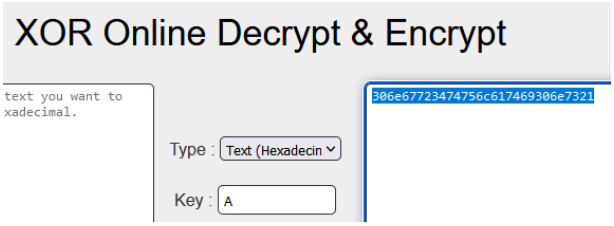
In loc_80491DE: it gives ‘i’ value of 0, then in loc_80491E5 it compares ‘i’ to input length. Loops next part is looks like XOR calculator. This was dead end for me with the loops, so I used chatgpt.

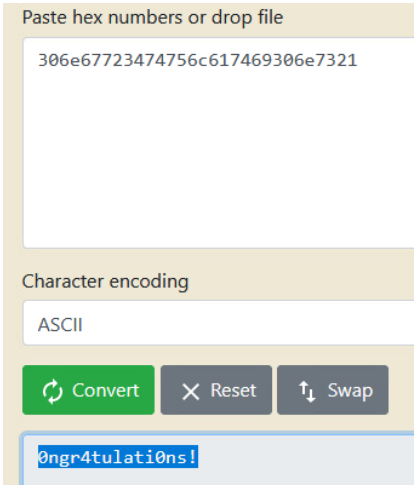


From chatgpt I get that XOR key is 65 which is A as a letter. And from PASSWORD I did find string ‘q/&3u54-5(q/2`’

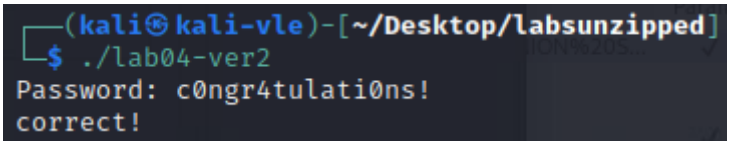
```
.rodata:0804A008 PASSWORD       db 22h                  ; DATA XREF: check_password+F↑o
.rodata:0804A008                                        ; check_password+64↑r
.rodata:0804A009 aQ3u545Q2      db 'q/&3u54- 5(q/2`',0
.rodata:0804A019 aPassword      db 'Password: ',0       ; DATA XREF: main+3A↑o
```

I decrypted it online with key A and get ‘306e67723474756c617469306e7321’ then I used hex to ASCII converter and get string ‘0ngr4tulati0ns!’





It seems that I'm on right path. Then I tried it and added c to start of string.



Right password is c0ngr4tulati0ns! but I don't have a clue where the c comes from?

When loop gets right password, it gives 'var_1' value 1 and moves it to al. At the end it jumps back to main and there it checks if password was correct and gives print 'correct'