Difficult 1

1 — What is the flow of the algorithm used to check the validity of a password?

1. The application XOR a memory zone with a constant (0x17 in my case).

```c
__addr[i] = (code)((&DAT_004020c0)[i] ^ 0x17);
```

2. The result of the XOR is put on the protected memory zone on the heap.

```c
__addr = (code *)valloc(1024); // Allocate memory aligned to page boundary
mprotect(__addr,1024,7); // protect this memory
```

3. It's cast on a function pointer to be executed with the password (*param_2[1]*) filled by the user as parameter.

```c
DAT_00404098 = (*__addr)(param_2[1]);
```

4. This newly "decrypted" function XOR each char of the password with a specific constant. The result as to be 0 for a valid password, so the password can be extracted as each constant.

```c
int check_password(char* password) {
  return (int)(char)( // Have to be equal to 0 for a good password
      password[0]  ^ 0x58 | // X
      password[1]  ^ 0x67 | // g
      password[2]  ^ 0x6e | // n
      password[3]  ^ 0x73 | // s
      password[4]  ^ 0x57 | // W
      password[5]  ^ 0x45 | // E
      password[6]  ^ 0x7a | // u
      password[7]  ^ 0x41 | // A
      password[8]  ^ 0x74 | // t
      password[9]  ^ 0x66 | // f
      password[10] ^ 0x42 | // B
      password[11] ^ 0x6f   // o
  );
}
```

2 — Can you recover the secret password? You must send 1 the valid password by email to pascal+sre25@mod-p.ch before Apr. 28th, 2025, 12h00 CET to validate this lab and get 30 points

XgnsWEuAtfBo

3 — Difficulties encountered during the lab

I didn't encounter particular difficulties during the lab.

I lose a bit of time trying to XOR directly data on Ghidra. I finally, export data from Ghidra, XOR them with a C script (Code 1) and export the result directly as a binary code in a file.
I finally opened this new file on Ghidra to decompile it.

```c
uint8_t super_function[] = { 0x5f, 0xaf, 0xac, 0x93, 0x9a, 0x87, 0xa3, 0xb1, 0x8e, 0xb5, 0x5f,
0x9e, 0x53, 0x33, 0xe7, 0xd1, 0x53, 0x33, 0xeb, 0x17, 0xd0, 0x53, 0x33, 0xef, 0x80, 0x92, 0xb6,
0x9b, 0x9d, 0x53, 0x33, 0xe7, 0x25, 0x10, 0x23, 0xf4, 0x9d, 0x5b, 0x33, 0xe6, 0x25, 0x58, 0x16,
0x97, 0xe6, 0xf4, 0x1f, 0xd6, 0x9d, 0x53, 0x33, 0xe5, 0x25, 0x50, 0x15, 0x23, 0xf4, 0x9d, 0x43,
0x33, 0xe4, 0x25, 0x40, 0x14, 0x97, 0xe5, 0xf4, 0x1f, 0xd5, 0x1f, 0xdd, 0x9d, 0x53, 0x33, 0xe3,
0x25, 0x50, 0x13, 0x23, 0xf4, 0x9d, 0x5b, 0x33, 0xe2, 0x25, 0x58, 0x12, 0x97, 0xe6, 0xf4, 0x1f,
0xd6, 0x9d, 0x53, 0x33, 0xe1, 0x25, 0x50, 0x11, 0x23, 0xf4, 0x1f, 0xdf, 0x9d, 0x5b, 0x33, 0xe0,
0x25, 0x58, 0x10, 0x1f, 0xc7, 0x9d, 0x43, 0x33, 0xef, 0x25, 0x40, 0x1f, 0x97, 0xe6, 0xf4, 0x97,
0xe5, 0xf4, 0x1f, 0xdd, 0x9d, 0x5b, 0x33, 0xee, 0x25, 0x58, 0x1e, 0x97, 0xe6, 0xf4, 0x1f, 0xc6,
0x9d, 0x43, 0x33, 0xed, 0x25, 0x40, 0x1d, 0x97, 0xe5, 0xf4, 0x1f, 0xdd, 0x9d, 0x5b, 0x33, 0xec,
0x25, 0x58, 0x1c, 0x1f, 0xd5, 0x97, 0xe6, 0xf4, 0x1f, 0xc6, 0x18, 0xa9, 0xd6, 0xd4 };

uint8_t main(int argc, char** argv) {
    uint8_t length = sizeof(super_function);
    uint8_t my_new_byte[length];
    for(uint8_t i = 0; i < length; i++) {
        my_new_byte[i] = super_function[i]^0x17;
    }

    FILE *fptr = fopen("./test_password", "wb+");
    if(fptr != NULL){
        fwrite(my_new_byte, 1, length, fptr);
        fclose(fptr);
    }
}
```

Code 1: Script to XOR data from Ghidra