Pico CTFs Write Ups week 4:

1. PW Crack 4:
    a. Downloaded the python program, encoded flag, and the hash file.
    b. Used the command "cat level4.py" to read the code and it uses hashes to compare the passwords as level3. It reads the hash value of the correct password and compares it with the password that the user inputs after passing it through a hashing function "hashlib.md5()", if its correct it returns the flag.
    c. Therefore, I used the command "xxd level4.hash.bin" and got the hexadecimal values "64c1ef0036a9a8d176fbc381c4abda52", I sent these values to an online hash md5 decoder and got the password "9f63" and this string was in the possible passwords given by the problem.
    d. Finally, I used the command "python3 level4.py", entered the password and got the flag.
    e. picoCTF{fl45h_5pr1ng1ng_d770d48c}
2. PW Crack 5:
    a. Downloaded the python program, encoded flag, hash file and dictionary.
    b. Used the command "cat leve5.py" to read the code and it's the same format as the previous level, however, it now includes a dictionary with all possible passwords. To get the password I created the program below to iterate through the passwords in dictionary and hash them until it found the one that is in the hash file. The password was "7e5f".
    c.
```
import hashlib

target = open(r"C:\Users\Jaime\Downloads\level5.hash.bin", "rb").read()

def hash_pw(pw_str):
    pw_bytes = bytearray()
    pw_bytes.extend(pw_str.encode())
    m = hashlib.md5()
    m.update(pw_bytes)
    return m.digest()

with open(r"C:\Users\Jaime\Downloads\dictionary.txt") as d:
    line = d.readline()
    while line:
        x = line.strip()
        x = hash_pw(x)
        if (x == target):
            print(line)
            break
        line = d.readline()
```
    d. Finally, I used the command "python3 level5.py", entered the password and got the flag.
    e. picoCTF{h45h_sl1ng1ng_40f26f81}
3. GDB baby step 4:
    a. Downloaded the program to debug and used the command "chmod +x debugger0_d" to make it executable.
    b. After that I used the command "gdb debugger0_d" and "disassemble main" to start debugging the program and get the assembly dump.

    c. Since the objective is to find the constant that is being used to multiply the value that is located in the register eax I used "break *main+36" and "break *main+43" to have breakpoints before and after the function is used.
    d. After using "run" the value of eax before the function was 654 and after 8439870.
    e. To get the constant I used a simple equation: 654(x) = 8439870 and the constant was 12905.
    f. picoCTF{12905}
4. ASCI FTW:
    a. Downloaded the program and used "chmod +x asciiftw" to make it executable.
    b. Then I used the command "gdb asciiftw" and "disassemble main" to get the assembly dump.
    c. After inspecting the code, the flag was hidden in hexadecimal numbers and moved to the stack, the numbers where "7b41534349495f49535f454153595f38393630463041467d"
    d. I used a hexadecimal to text converter, and it resulted in "{ASCII_IS_EASY_8960F0AF}" and got the flag.
    e. picoCTF{ASCII_IS_EASY_8960F0AF}
5. Picker II:
    a. Downloaded the python program and used "cat picker-II.py" to read the code and notices that it used "eval" in a while true loop.
    b. In addition, the flag was stored in a file named "flag.txt" so I used the command "print(open('flag.txt','r').read())" and got the flag.
    c. picoCTF{f1l73r5_f41l_c0d3_r3f4c70r_m1gh7_5ucc33d_95d44590}