Открываю программу в Ghidra. Смотрю определения строк, нахожу те, за которые можно зацепиться

140003418	Enter a username!	"Enter a username!"	ds
140003430	Enter a key!	"Enter a key!"	ds
140003440	Access Granted!	"Access Granted!"	ds
140003aa8	Incorrect Key...	"Incorrect Key..."	ds
140003ac0	enter 0 to exit or 1 to try again	"enter 0 to exit or 1 to try again"	ds

Перехожу по xref-адресам:

Думаю, в первую очередь меня интересует условие успешного завершение, так как неверный ввод выкинут в ветку else

```
pbVar3 = FUN_140001700((basic_ostream<> *)cout_exref,"Enter a username!");
std::basic_ostream<>::operator<<(pbVar3,FUN_1400018d0);
FUN_140001910((basic_istream<> *)cin_exref,(longlong *)&username_input,param_3);
pbVar3 = FUN_140001700((basic_ostream<> *)cout_exref,"Enter a key!");
std::basic_ostream<>::operator<<(pbVar3,FUN_1400018d0);
FUN_140001910((basic_istream<> *)cin_exref,(longlong *)&key_input,param_3);
uVar2 = local_18;
pKey = key_input;
uVar1 = local_38;
pLogin = username_input;
cheksum = 0;
uVar4 = uVar10;
uVar7 = uVar10;
uVar9 = uVar10;
if (local_40 != 0) {
  do {
    ppppuVar5 = &username_input;
    if (0xf < local_38) {
      ppppuVar5 = (undefined8 ****)username_input;
    }
    cheksum = (int)uVar9 + (int)*(char *)(uVar4 + (longlong)ppppuVar5);
    uVar9 = (ulonglong)cheksum;
    uVar6 = (int)uVar7 + 1;
    uVar7 = (ulonglong)uVar6;
    uVar4 = uVar4 + 1;
  } while ((ulonglong)(longlong)(int)uVar6 < local_40);
}
cheksum = (int)cheksum % 10;
param_3 = (ulonglong)cheksum;
if ((uint)local_20 == cheksum) {
  ppppuVar5 = &key_input;
  if (0xf < local_18) {
    ppppuVar5 = (undefined8 ****)key_input;
  }
  if (*(char *)((longlong)(int)(uint)local_20 + -1 + (longlong)ppppuVar5) !=
      (char)((char)cheksum + '0')) goto LAB_140001413;
  pbVar3 = FUN_140001700((basic_ostream<> *)cout_exref,"Access Granted!");
  std::basic_ostream<>::operator<<(pbVar3,FUN_1400018d0);
  pcVar8 =
  "\n                                        $$$$$$$$$$$$$$$$$$$$\n                          $$$$$$
  $$$$$$$$$$$$$$$$$$$$$\n                        $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$         $$   $$$$$
```

Начинаю смотреть и думать.

Это похоже на символьную декомпозицию введенного логина

```
do {
  pppuVar5 = &username_input;
  if (0xf < local_38) {
    pppuVar5 = (undefined8 ****)username_input;
  }
  cheksum = (int)uVar9 + (int)*(char *)(uVar4 + (longlong)pppuVar5);
  uVar9 = (ulonglong)cheksum;
  uVar6 = (int)uVar7 + 1;
  uVar7 = (ulonglong)uVar6;
  uVar4 = uVar4 + 1;
} while ((ulonglong)(longlong)(int)uVar6 < local_40);
```

Причем тут суммируется ASCII-код каждого символа. Значит, для логина необходимо вычислить сумму аски-кодов составляющих его букв

А дальше, судя по всему, идет проверка пароля

```
cheksum = (int)cheksum % 10;
param_3 = (ulonglong)cheksum;
if ((uint)local_20 == cheksum) {
  pppuVar5 = &key_input;
  if (0xf < local_18) {
    pppuVar5 = (undefined8 ****)key_input;
  }
  if (*(char *)((longlong)(int)(uint)local_20 + -1 + (longlong)pppuVar5) !=
      (char)((char)cheksum + '0')) goto LAB_140001413;
  pbVar3 = FUN_140001700((basic_ostream<> *)cout_exref,"Access Granted!");
  std::basic_ostream<>::operator<<(pbVar3,FUN_1400018d0);
```

Берется последняя цифра суммы кодов символов логина, и она должна быть как длиной пароля, так и последним его символом.

Проверим. Напишем простой скрипт для вычисления контрольной суммы

```
username = input()
sum = 0
for char in username:
    sum+=ord(char)
checksum = sum %10
print(sum)
print(checksum)
```

```
user
447
7
```

Пробуем: