Running the executable

```
root@kali:~/cyberedu/reverse# chmod +x fake-add
root@kali:~/cyberedu/reverse# ./fake-add
[+] Solve all additions and find the flag!!
root@kali:~/cyberedu/reverse#
```

Opening main function in IDA

```
; Attributes: bp-based frame

; int __fastcall main(int, char **, char **)
main proc near
; __unwind {
endbr64
push    rbp
mov     rbp, rsp
call    sub_11A9
mov     eax, 0
pop     rbp
retn
; } // starts at 14AD
main endp
```

There is a call to another function called “sub_11A9”.

```
endbr64
push    rbp
mov     rbp, rsp
sub     rsp, 100h
mov     [rbp+var_FC], 3Ch ; '<'
mov     [rbp+var_F8], 7
mov     [rbp+var_F4], 2Ah ; '*'
mov     [rbp+var_F0], 2Ah ; '*'
mov     [rbp+var_EC], 20h ; ' '
mov     [rbp+var_E8], 26h ; '&'
mov     [rbp+var_E4], 78h ; 'x'
mov     [rbp+var_E0], 3
mov     [rbp+var_DC], 5Ah ; 'Z'
mov     [rbp+var_D8], 1Ah
mov     [rbp+var_D4], 68h ; 'h'
mov     [rbp+var_D0], 0
mov     [rbp+var_CC], 27h ; '''
mov     [rbp+var_C8], 0Ah
mov     [rbp+var_C4], 64h ; 'd'
mov     [rbp+var_C0], 0Fh
mov     [rbp+var_BC], 4Bh ; 'K'
mov     [rbp+var_B8], 14h
mov     [rbp+var_B4], 5Fh ; '_'
mov     [rbp+var_B0], 0Ah
mov     [rbp+var_AC], 64h ; 'd'
mov     [rbp+var_A8], 0Fh
mov     [rbp+var_A4], 55h ; 'U'
mov     [rbp+var_A0], 0Ah
mov     [rbp+var_9C], 55h ; 'U'
mov     [rbp+var_98], 15h
mov     [rbp+var_94], 55h ; 'U'
mov     [rbp+var_90], 20h ; ' '
mov     [rbp+var_8C], 34h ; '4'
mov     [rbp+var_88], 1
mov     [rbp+var_84], 2Ah ; '*'
mov     [rbp+var_80], 2Ah ; '*'
mov     [rbp+var_7C], 35h ; '5'
mov     [rbp+var_78], 2Ah ; '*'
mov     [rbp+var_74], 21h ; '!'
mov     [rbp+var_70], 20h ; ' '
mov     [rbp+var_6C], 21h ; '!'
```

Looks like it is declaring variables and giving them values.

And down we can see it is adding those variables.

```
mov     [rbp+var_50], eax
mov     edx, [rbp+var_EC]
mov     eax, [rbp+var_E8]
add     eax, edx
mov     [rbp+var_4C], eax
mov     edx, [rbp+var_E4]
mov     eax, [rbp+var_E0]
add     eax, edx
mov     [rbp+var_48], eax
mov     edx, [rbp+var_DC]
mov     eax, [rbp+var_D8]
add     eax, edx
mov     [rbp+var_44], eax
mov     edx, [rbp+var_D4]
mov     eax, [rbp+var_D0]
add     eax, edx
mov     [rbp+var_40], eax
mov     edx, [rbp+var_CC]
mov     eax, [rbp+var_C8]
add     eax, edx
mov     [rbp+var_3C], eax
mov     edx, [rbp+var_C4]
mov     eax, [rbp+var_C0]
add     eax, edx
mov     [rbp+var_38], eax
mov     edx, [rbp+var_BC]
mov     eax, [rbp+var_B8]
add     eax, edx
```

To solve this, we must replicate what's happening here to see the result.

Let's write it in c++.

```
// Perform the additions
int var_54 = var_FC + var_F8;
int var_50 = var_F4 + var_F0;
int var_4C = var_EC + var_E8;
int var_48 = var_E4 + var_E0;
int var_44 = var_DC + var_D8;
int var_40 = var_D4 + var_D0;
int var_3C = var_CC + var_C8;
int var_38 = var_C4 + var_C0;
int var_34 = var_BC + var_B8;
int var_30 = var_B4 + var_B0;
int var_2C = var_AC + var_A8;
int var_28 = var_A4 + var_A0;
int var_24 = var_9C + var_98;
int var_20 = var_94 + var_90;
int var_1C = var_8C + var_88;
int var_18 = var_84 + var_80;
int var_14 = var_7C + var_78;
int var_10 = var_74 + var_70;
int var_C = var_6C + var_68;
int var_8 = var_64 + var_60;
int var_4 = var_5C + var_58;
```

I will upload the code.

```
67 84 70 123 116 104 49 115 95 105 115 95 106 117 53 84 95 65 68 68 125
Process returned 0 (0x0)   execution time : 0.041 s
Press any key to continue.
```

Looks like decimal.

And the flag

Output

```
CTF{th1s_is_ju5T_ADD}
```