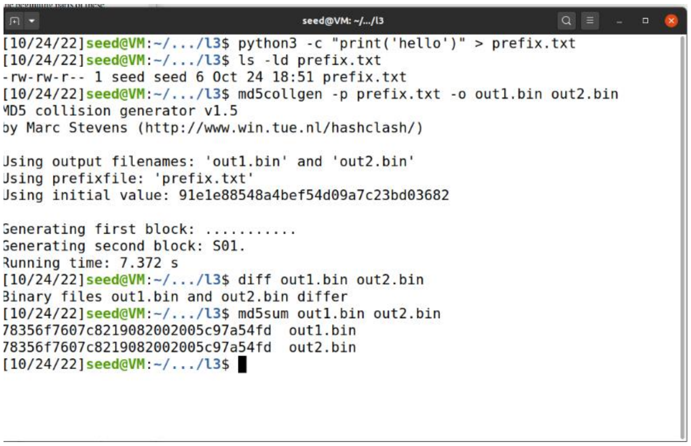
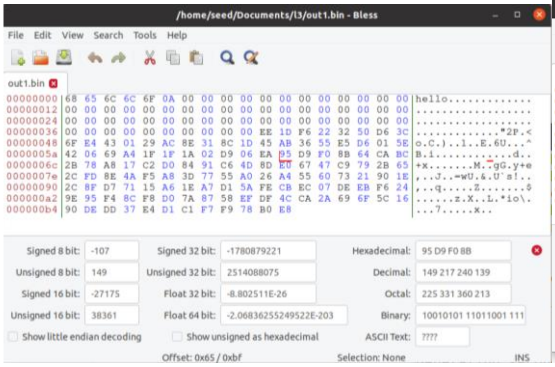
Task1



seed@VM: ~/.../l3

```
[10/24/22]seed@VM:~/.../l3$ python3 -c "print('hello')" > prefix.txt
[10/24/22]seed@VM:~/.../l3$ ls -ld prefix.txt
-rw-rw-r-- 1 seed seed 6 Oct 24 18:51 prefix.txt
[10/24/22]seed@VM:~/.../l3$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 91e1e88548a4bef54d09a7c23bd03682

Generating first block: ...........
Generating second block: S01.
Running time: 7.372 s
[10/24/22]seed@VM:~/.../l3$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[10/24/22]seed@VM:~/.../l3$ md5sum out1.bin out2.bin
78356f7607c8219082002005c97a54fd  out1.bin
78356f7607c8219082002005c97a54fd  out2.bin
[10/24/22]seed@VM:~/.../l3$ 
```

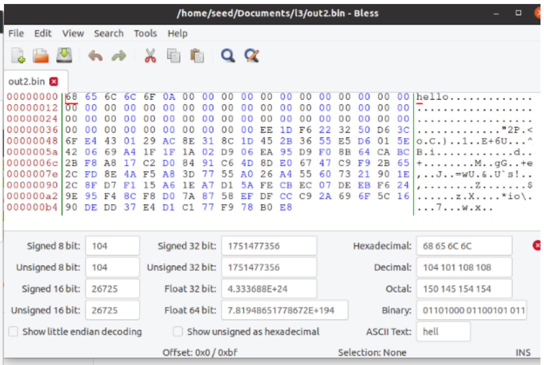
First, I follow the instruction to build to two files who have the same MD5 hash value and save them into out1.bin and out2.bin. Using “diff out1.bin out2.bin” command, the output shows that they have the different content. I use binary editor to view them, and it can be noticed that the output is different. At the same time, the output of md5sum is same, which means they have the same hash value.



/home/seed/Documents/l3/out1.bin - Bless

```
00000000 68 65 6C 6C 6F 0A 00 00 00 00 00 00 00 00 00 00 00 00 hello.............
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 EE 1D F6 22 32 50 D6 3C ............."2P.<
00000048 6F E4 43 01 29 AC 8E 31 8C 1D 45 AB 36 55 E5 D6 01 5E o.C.)..1..E.6U...^
0000005a 42 06 69 A4 1F 1F 1A 02 D9 06 EA 95 D9 F0 8B 64 CA BC B.i.............d..
0000006c 2B 78 A8 17 C2 D0 84 91 C6 4D 8D E0 67 47 C9 79 2B 65 +x.......M..gG.y+e
0000007e 2C FD 8E 4A F5 A8 3D 77 55 A0 26 A4 55 60 73 21 90 1E ,..J..=wU.&.U`s!..
00000090 2C 8F D7 71 15 A6 1E A7 D1 5A FE CB EC 07 DE EB F6 24 ,..q.....Z.......$
000000a2 9E 95 F4 8C F8 D0 7A 87 58 EF DF 4C CA 2A 69 6F 5C 16 ......z.X..L.*io\.
000000b4 90 DE DD 37 E4 D1 C1 F7 F9 78 B0 E8                   ...7.....x..
```

Signed 8 bit: -107 Signed 32 bit: -1780879221 Hexadecimal: 95 D9 F0 8B
Unsigned 8 bit: 149 Unsigned 32 bit: 2514088075 Decimal: 149 217 240 139
Signed 16 bit: -27175 Float 32 bit: -8.802511E-26 Octal: 225 331 360 213
Unsigned 16 bit: 38361 Float 64 bit: -2.06836255249522E-203 Binary: 10010101 11011001 111
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: ????
Offset: 0x65 / 0xbf Selection: None INS

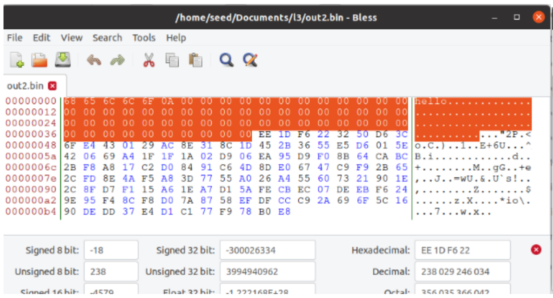


/home/seed/Documents/l3/out2.bin - Bless

```
00000000 68 65 6C 6C 6F 0A 00 00 00 00 00 00 00 00 00 00 00 00 hello.............
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 EE 1D F6 22 32 50 D6 3C ............."2P.<
00000048 6F E4 43 01 29 AC 8E 31 8C 1D 45 2B 36 55 E5 D6 01 5E o.C.)..1..E+6U...^
0000005a 42 06 69 A4 1F 1F 1A 02 D9 06 EA 95 D9 F0 8B 64 CA BC B.i.............d..
0000006c 2B F8 A8 17 C2 D0 84 91 C6 4D 8D E0 67 47 C9 F9 2B 65 +........M..gG..+e
0000007e 2C FD 8E 4A F5 A8 3D 77 55 A0 26 A4 55 60 73 21 90 1E ,..J..=wU.&.U`s!..
00000090 2C 8F D7 F1 15 A6 1E A7 D1 5A FE CB EC 07 DE EB F6 24 ,.........Z.......$
000000a2 9E 95 F4 8C F8 D0 7A 87 58 EF DF CC C9 2A 69 6F 5C 16 ......z.X....*io\.
000000b4 90 DE DD 37 E4 D1 C1 77 F9 78 B0 E8                   ...7...w.x..
```

Signed 8 bit: 104 Signed 32 bit: 1751477356 Hexadecimal: 68 65 6C 6C
Unsigned 8 bit: 104 Unsigned 32 bit: 1751477356 Decimal: 104 101 108 108
Signed 16 bit: 26725 Float 32 bit: 4.333688E+24 Octal: 150 145 154 154
Unsigned 16 bit: 26725 Float 64 bit: 7.81948651778672E+194 Binary: 01101000 01100101 011
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: hell
Offset: 0x0 / 0xbf Selection: None INS

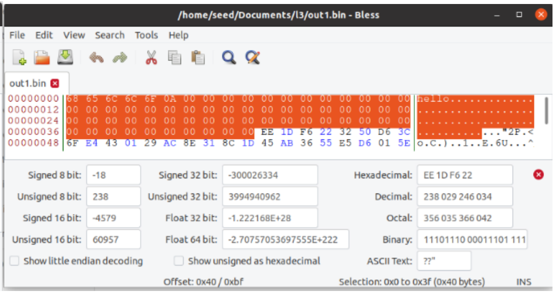
Q1:



/home/seed/Documents/l3/out2.bin - Bless

```
00000000 68 65 6C 6C 6F 0A 00 00 00 00 00 00 00 00 00 00 00 00 hello.............
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 EE 1D F6 22 32 50 D6 3C ............."2P.<
00000048 6F E4 43 01 29 AC 8E 31 8C 1D 45 2B 36 55 E5 D6 01 5E o.C.)..1..E+6U...^
0000005a 42 06 69 A4 1F 1F 1A 02 D9 06 EA 95 D9 F0 8B 64 CA BC B.i.............d..
0000006c 2B F8 A8 17 C2 D0 84 91 C6 4D 8D E0 67 47 C9 F9 2B 65 +........M..gG..+e
0000007e 2C FD 8E 4A F5 A8 3D 77 55 A0 26 A4 55 60 73 21 90 1E ,..J..=wU.&.U`s!..
00000090 2C 8F D7 F1 15 A6 1E A7 D1 5A FE CB EC 07 DE EB F6 24 ,.........Z.......$
000000a2 9E 95 F4 8C F8 D0 7A 87 58 EF DF CC C9 2A 69 6F 5C 16 ......z.X....*io\.
000000b4 90 DE DD 37 E4 D1 C1 77 F9 78 B0 E8                   ...7...w.x..
```

Signed 8 bit: -18 Signed 32 bit: -300026334 Hexadecimal: EE 1D F6 22
Unsigned 8 bit: 238 Unsigned 32 bit: 3994940962 Decimal: 238 029 246 034
Signed 16 bit: -4579 Float 32 bit: -1.222168E+28 Octal: 356 035 366 042



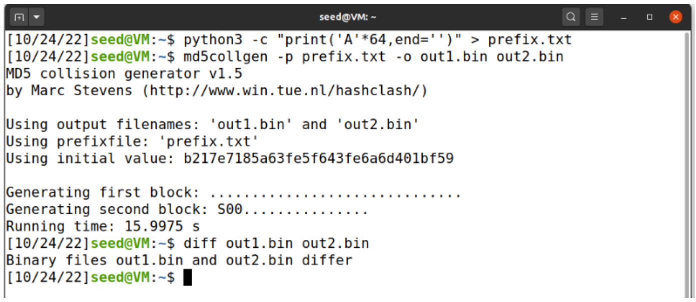
/home/seed/Documents/l3/out1.bin - Bless

```
00000000 68 65 6C 6C 6F 0A 00 00 00 00 00 00 00 00 00 00 00 00 hello.............
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 EE 1D F6 22 32 50 D6 3C ............."2P.<
00000048 6F E4 43 01 29 AC 8E 31 8C 1D 45 AB 36 55 E5 D6 01 5E o.C.)..1..E.6U...^
```

Signed 8 bit: -18 Signed 32 bit: -300026334 Hexadecimal: EE 1D F6 22
Unsigned 8 bit: 238 Unsigned 32 bit: 3994940962 Decimal: 238 029 246 034
Signed 16 bit: -4579 Float 32 bit: -1.222168E+28 Octal: 356 035 366 042
Unsigned 16 bit: 60957 Float 64 bit: -2.70757053697555E+222 Binary: 11101110 00011101 111
☐ Show little endian decoding ☐ Show unsigned as hexadecimal ASCII Text: ??"
Offset: 0x40 / 0xbf Selection: 0x0 to 0x3f (0x40 bytes) INS

If the length of your prefix file is not multiple of 64, the space will be padded by 0 to make sure the prefix file size are the multiple of 64. It can be observed at the pictures as above.

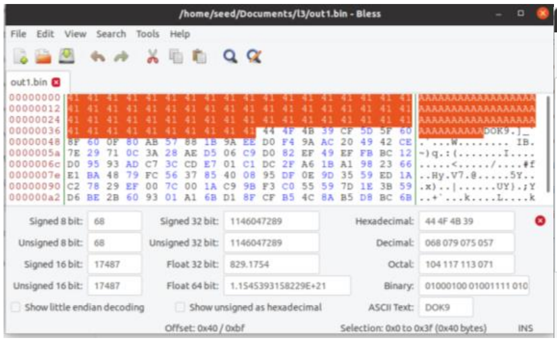
Q2:



seed@VM: ~

```
[10/24/22]seed@VM:~$ python3 -c "print('A'*64,end='')" > prefix.txt
[10/24/22]seed@VM:~$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: b217e7185a63fe5f643fe6a6d401bf59

Generating first block: .............................
Generating second block: S00...............
Running time: 15.9975 s
[10/24/22]seed@VM:~$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[10/24/22]seed@VM:~$ 
```

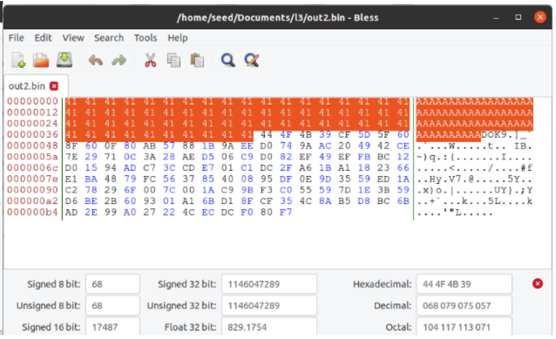
Build two files with 64 bytes prefix file. And the outputs shown as follows. It can be noticed that prefix file is not padded because the file size is 64 bytes.



/home/seed/Documents/l3/out1.bin - Bless

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 41 44 4F 4B 39 CF 5D 5F 60 AAAAAAAAAADOK9.]_`
00000048 8F 60 0F 80 AB 57 88 1B 9A EE D0 F4 9A AC 20 49 42 CE .`...W........ IB.
0000005a 7E 29 71 0C 3A 28 AE D5 06 C9 D0 82 EF 49 EF FB BC 12 ~)q.:(.......I....
0000006c D0 95 93 AD C7 3C CD E7 01 C1 DC 2F A6 1B A1 98 23 66 .....<...../....#f
0000007e E1 BA 48 79 FC 56 37 85 40 08 95 DF 0E 9D 35 59 ED 1A ..Hy.V7.@.....5Y..
00000090 C2 78 29 EF 00 7C 00 1A C9 9B F3 C0 55 59 7D 1E 3B 59 .x)..|......UY}.;Y
000000a2 D6 BE 2B 60 93 01 A1 6B D1 8F CF B5 4C 8A B5 D8 BC 6B ..+`...k....L....k
```

```
Signed 8 bit: 68        Signed 32 bit: 1146047289         Hexadecimal: 44 4F 4B 39
Unsigned 8 bit: 68      Unsigned 32 bit: 1146047289       Decimal: 068 079 075 057
Signed 16 bit: 17487    Float 32 bit: 829.1754            Octal: 104 117 113 071
Unsigned 16 bit: 17487  Float 64 bit: 1.1545393158229E+21 Binary: 01000100 01001111 010
□ Show little endian decoding  □ Show unsigned as hexadecimal  ASCII Text: DOK9
Offset: 0x40 / 0xbf    Selection: 0x0 to 0x3f (0x40 bytes)    INS
```

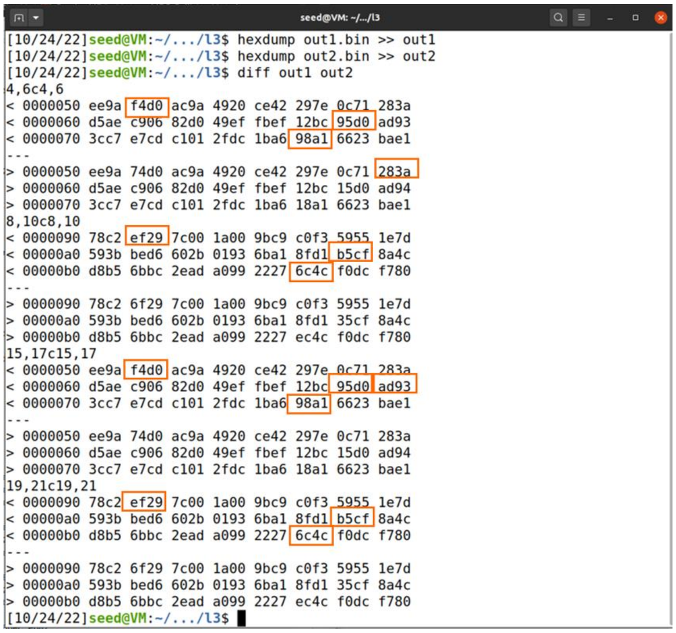


/home/seed/Documents/l3/out2.bin - Bless

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 41 44 4F 4B 39 CF 5D 5F 60 AAAAAAAAAADOK9.]_`
00000048 8F 60 0F 80 AB 57 88 1B 9A EE D0 74 9A AC 20 49 42 CE .`...W.....t.. IB.
0000005a 7E 29 71 0C 3A 28 AE D5 06 C9 D0 82 EF 49 EF FB BC 12 ~)q.:(.......I....
0000006c D0 15 94 AD C7 3C CD E7 01 C1 DC 2F A6 1B A1 18 23 66 .....<...../....#f
0000007e E1 BA 48 79 FC 56 37 85 40 08 95 DF 0E 9D 35 59 ED 1A ..Hy.V7.@.....5Y..
00000090 C2 78 29 6F 00 7C 00 1A C9 9B F3 C0 55 59 7D 1E 3B 59 .x)o.|......UY}.;Y
000000a2 D6 BE 2B 60 93 01 A1 6B D1 8F CF 35 4C 8A B5 D8 BC 6B ..+`...k...5L....k
000000b4 AD 2E 99 A0 27 22 4C EC DC F0 80 F7                   ....'"L.....
```

```
Signed 8 bit: 68        Signed 32 bit: 1146047289         Hexadecimal: 44 4F 4B 39
Unsigned 8 bit: 68      Unsigned 32 bit: 1146047289       Decimal: 068 079 075 057
Signed 16 bit: 17487    Float 32 bit: 829.1754            Octal: 104 117 113 071
```

Q3:

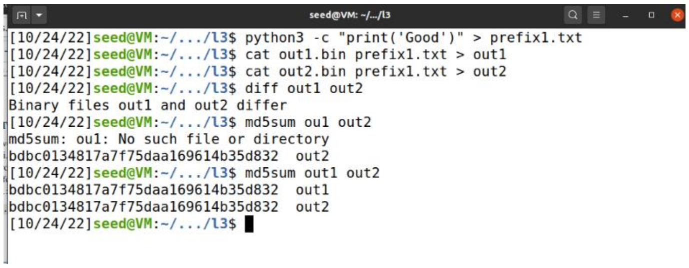


seed@VM: ~/.../l3

```
[10/24/22]seed@VM:~/.../l3$ hexdump out1.bin >> out1
[10/24/22]seed@VM:~/.../l3$ hexdump out2.bin >> out2
[10/24/22]seed@VM:~/.../l3$ diff out1 out2
4,6c4,6
< 0000050 ee9a f4d0 ac9a 4920 ce42 297e 0c71 283a
< 0000060 d5ae c906 82d0 49ef fbef 12bc 95d0 ad93
< 0000070 3cc7 e7cd c101 2fdc 1ba6 98a1 6623 bae1
---
> 0000050 ee9a 74d0 ac9a 4920 ce42 297e 0c71 283a
> 0000060 d5ae c906 82d0 49ef fbef 12bc 15d0 ad94
> 0000070 3cc7 e7cd c101 2fdc 1ba6 18a1 6623 bae1
8,10c8,10
< 0000090 78c2 ef29 7c00 1a00 9bc9 c0f3 5955 1e7d
< 00000a0 593b bed6 602b 0193 6ba1 8fd1 b5cf 8a4c
< 00000b0 d8b5 6bbc 2ead a099 2227 6c4c f0dc f780
---
> 0000090 78c2 6f29 7c00 1a00 9bc9 c0f3 5955 1e7d
> 00000a0 593b bed6 602b 0193 6ba1 8fd1 35cf 8a4c
> 00000b0 d8b5 6bbc 2ead a099 2227 ec4c f0dc f780
15,17c15,17
< 0000050 ee9a f4d0 ac9a 4920 ce42 297e 0c71 283a
< 0000060 d5ae c906 82d0 49ef fbef 12bc 95d0 ad93
< 0000070 3cc7 e7cd c101 2fdc 1ba6 98a1 6623 bae1
---
> 0000050 ee9a 74d0 ac9a 4920 ce42 297e 0c71 283a
> 0000060 d5ae c906 82d0 49ef fbef 12bc 15d0 ad94
> 0000070 3cc7 e7cd c101 2fdc 1ba6 18a1 6623 bae1
19,21c19,21
< 0000090 78c2 ef29 7c00 1a00 9bc9 c0f3 5955 1e7d
< 00000a0 593b bed6 602b 0193 6ba1 8fd1 b5cf 8a4c
< 00000b0 d8b5 6bbc 2ead a099 2227 6c4c f0dc f780
---
> 0000090 78c2 6f29 7c00 1a00 9bc9 c0f3 5955 1e7d
> 00000a0 593b bed6 602b 0193 6ba1 8fd1 35cf 8a4c
> 00000b0 d8b5 6bbc 2ead a099 2227 ec4c f0dc f780
[10/24/22]seed@VM:~/.../l3$ 
```

It is not complete different, use diff command to check the different parts. The different can be noticed as above.

Task 2



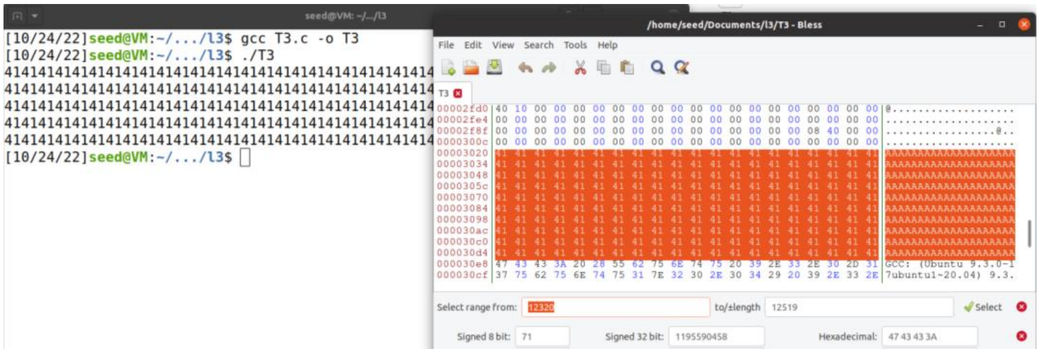
```
[10/24/22]seed@VM:~/.../l3$ python3 -c "print('Good')" > prefix1.txt
[10/24/22]seed@VM:~/.../l3$ cat out1.bin prefix1.txt > out1
[10/24/22]seed@VM:~/.../l3$ cat out2.bin prefix1.txt > out2
[10/24/22]seed@VM:~/.../l3$ diff out1 out2
Binary files out1 and out2 differ
[10/24/22]seed@VM:~/.../l3$ md5sum oul out2
md5sum: oul: No such file or directory
bdbc0134817a7f75daa169614b35d832  out2
[10/24/22]seed@VM:~/.../l3$ md5sum out1 out2
bdbc0134817a7f75daa169614b35d832  out1
bdbc0134817a7f75daa169614b35d832  out2
[10/24/22]seed@VM:~/.../l3$ 
```

I built out1.bin and out2.bin in task 1, and MD5(out1.bin) == MD5(out2.bin). I build a new file and cat it with out1.bin and out2. bin. Then I observed that the MD5 value also same. It can be proofed that the hash value will also be same after two files have the same hash value concatenate same content.

Task 3

```
#include <stdio.h>
unsigned char xyz[200] = {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"};
int main(){
  int i;
  for (i = 0; i < 200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}
```

I modify the C code as above, and gcc it gets T3. Open T3 file and can be find the array are saving at 0x3020 that the decimal is 12320.



```
[10/24/22]seed@VM:~/.../l3$ gcc T3.c -o T3
[10/24/22]seed@VM:~/.../l3$ ./T3
414141414141414141414141414141414141414141414141414
414141414141414141414141414141414141414141414141414
414141414141414141414141414141414141414141414141414
414141414141414141414141414141414141414141414141414
414141414141414141414141414141414141414141414141414
[10/24/22]seed@VM:~/.../l3$ 
```

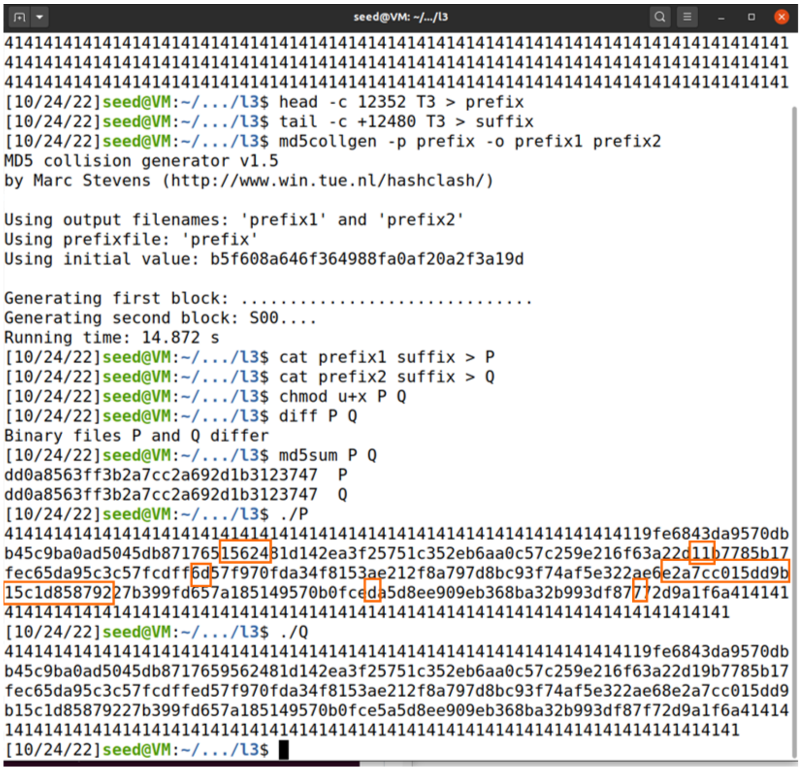
/home/seed/Documents/l3/T3 - Bless

```
00002fd0 40 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 @...................
00002fe4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....................
00002ff8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 00 00 ................@..
0000300c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....................
00003020 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
00003034 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
00003048 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
0000305c 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
00003070 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
00003084 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
00003098 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
000030ac 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
000030c0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
000030d4 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAA
000030e8 47 43 43 3A 20 28 55 62 75 6E 74 75 20 39 2E 33 2E 30 2D 31 GCC: (Ubuntu 9.3.0-1
000030cf 37 75 62 75 6E 74 75 31 7E 32 30 2E 30 34 29 20 39 2E 33 2E 7ubuntu1~20.04) 9.3.
```

Select range from: 12320 to/±length 12519

Signed 8 bit: 71 Signed 32 bit: 1195590458 Hexadecimal: 47 43 43 3A

We need the size of prefix file is multiply 64 bytes, and 12352 can be divided by 64. Therefore, I split the first 12352 bytes to prefix, and split from the 12352+128 bytes to the end of the file to suffix. I use prefix to build two files "prefix1" and "prefix2" that have the same MD5 hash value. Then cat prefix1 and suffix to P, cat prefix2 and suffix to Q. Finally, verify P and Q. The hash value of P and Q are same, and P and Q can be executable. However, their outputs are not same.



```
41414141414141414141414141414141414141414141414141414141414141414141414141414141
41414141414141414141414141414141414141414141414141414141414141414141414141414141
41414141414141414141414141414141414141414141414141414141414141414141414141414141
[10/24/22]seed@VM:~/.../l3$ head -c 12352 T3 > prefix
[10/24/22]seed@VM:~/.../l3$ tail -c +12480 T3 > suffix
[10/24/22]seed@VM:~/.../l3$ md5collgen -p prefix -o prefix1 prefix2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'prefix1' and 'prefix2'
Using prefixfile: 'prefix'
Using initial value: b5f608a646f364988fa0af20a2f3a19d

Generating first block: .............................
Generating second block: S00....
Running time: 14.872 s
[10/24/22]seed@VM:~/.../l3$ cat prefix1 suffix > P
[10/24/22]seed@VM:~/.../l3$ cat prefix2 suffix > Q
[10/24/22]seed@VM:~/.../l3$ chmod u+x P Q
[10/24/22]seed@VM:~/.../l3$ diff P Q
Binary files P and Q differ
[10/24/22]seed@VM:~/.../l3$ md5sum P Q
dd0a8563ff3b2a7cc2a692d1b3123747  P
dd0a8563ff3b2a7cc2a692d1b3123747  Q
[10/24/22]seed@VM:~/.../l3$ ./P
414141414141414141414141414141414141414141414141414141414141414119fe6843da9570db
b45c9ba0ad5045db871765156248 1d142ea3f25751c352eb6aa0c57c259e216f63a22d11b7785b17
fec65da95c3c57fcdff6d57f970fda34f8153ae212f8a797d8bc93f74af5e322ae6e2a7cc015dd9b
15c1d85879227b399fd657a185149570b0fceda5d8ee909eb368ba32b993df87772d9a1f6a414141
41414141414141414141414141414141414141414141414141414141414141414141414141
[10/24/22]seed@VM:~/.../l3$ ./Q
414141414141414141414141414141414141414141414141414141414141414119fe6843da9570db
b45c9ba0ad5045db87176595624 81d142ea3f25751c352eb6aa0c57c259e216f63a22d19b7785b17
fec65da95c3c57fcdffed57f970fda34f8153ae212f8a797d8bc93f74af5e322ae68e2a7cc015dd9
b15c1d85879227b399fd657a185149570b0fce5a5d8ee909eb368ba32b993df87f72d9a1f6a41414
141414141414141414141414141414141414141414141414141414141414141414141414141
[10/24/22]seed@VM:~/.../l3$ 
```

T4

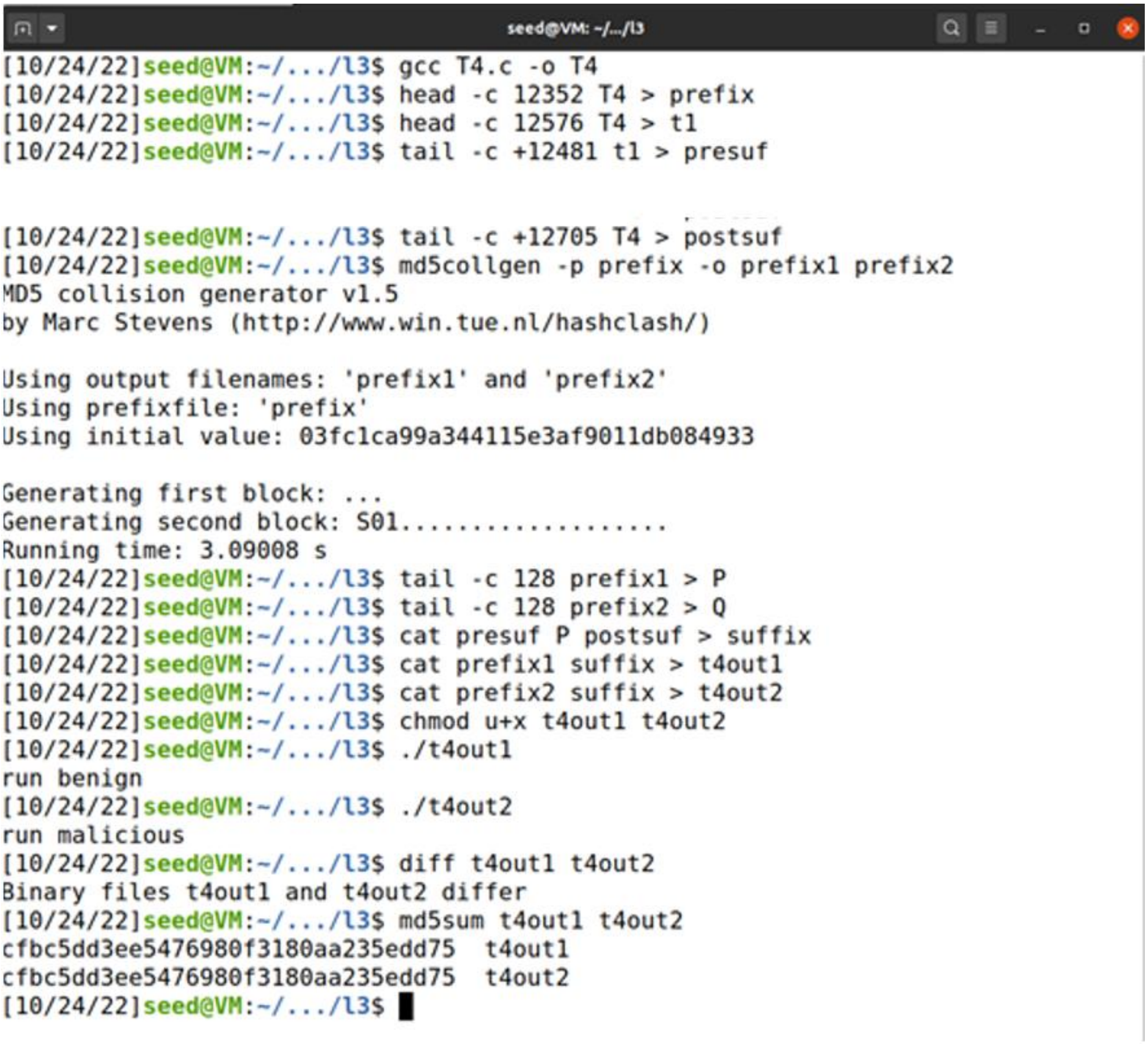
#include <stdio.h>

unsigned char X[200] = {

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

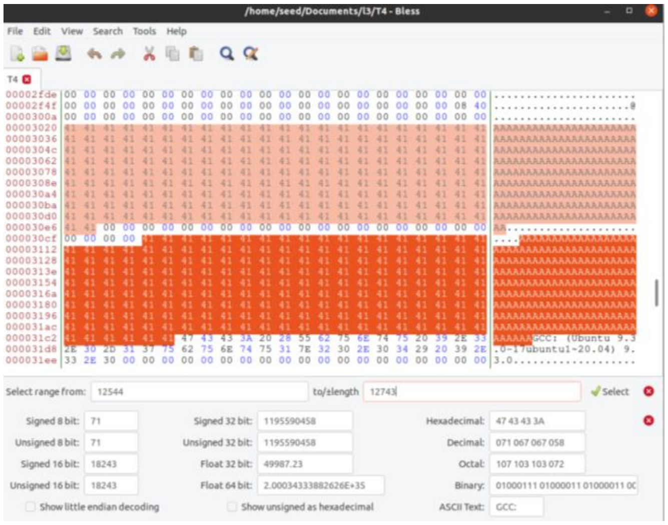
```
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    };

unsigned char Y[200] ={
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
    };

int main(){
    int t =1;
    for (int i = 0; i < 200; i++){
        if (X[i] != Y[i]){
            t = 0;
        }
    }
    if(t==0){
            printf("run malicious\n");
    }else{
            printf("run benign\n");
    }
    return 0;
}
```



```
[10/24/22]seed@VM:~/.../l3$ gcc T4.c -o T4
[10/24/22]seed@VM:~/.../l3$ head -c 12352 T4 > prefix
[10/24/22]seed@VM:~/.../l3$ head -c 12576 T4 > t1
[10/24/22]seed@VM:~/.../l3$ tail -c +12481 t1 > presuf


[10/24/22]seed@VM:~/.../l3$ tail -c +12705 T4 > postsuf
[10/24/22]seed@VM:~/.../l3$ md5collgen -p prefix -o prefix1 prefix2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'prefix1' and 'prefix2'
Using prefixfile: 'prefix'
Using initial value: 03fc1ca99a344115e3af9011db084933

Generating first block: ...
Generating second block: S01...................
Running time: 3.09008 s
[10/24/22]seed@VM:~/.../l3$ tail -c 128 prefix1 > P
[10/24/22]seed@VM:~/.../l3$ tail -c 128 prefix2 > Q
[10/24/22]seed@VM:~/.../l3$ cat presuf P postsuf > suffix
[10/24/22]seed@VM:~/.../l3$ cat prefix1 suffix > t4out1
[10/24/22]seed@VM:~/.../l3$ cat prefix2 suffix > t4out2
[10/24/22]seed@VM:~/.../l3$ chmod u+x t4out1 t4out2
[10/24/22]seed@VM:~/.../l3$ ./t4out1
run benign
[10/24/22]seed@VM:~/.../l3$ ./t4out2
run malicious
[10/24/22]seed@VM:~/.../l3$ diff t4out1 t4out2
Binary files t4out1 and t4out2 differ
[10/24/22]seed@VM:~/.../l3$ md5sum t4out1 t4out2
cfbc5dd3ee5476980f3180aa235edd75  t4out1
cfbc5dd3ee5476980f3180aa235edd75  t4out2
[10/24/22]seed@VM:~/.../l3$ █
```

This is the whole process of Task4.

Compile T4.c and open T4. Find Array X and Array Y. It can be noticed that Array X starts at 12320 Bytes, Array Y starts at 12544 Bytes.



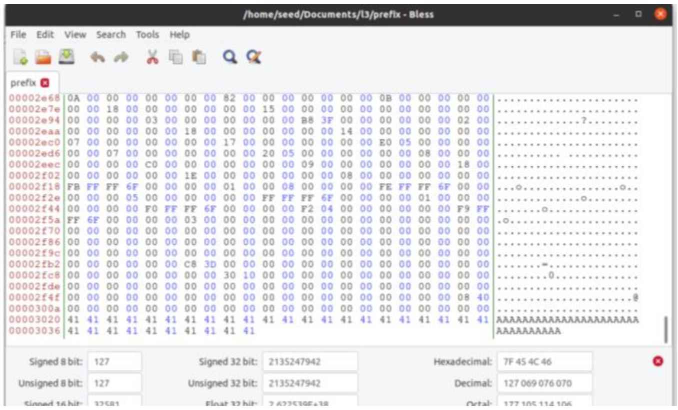
/home/seed/Documents/l3/T4 - Bless

```
00002fde 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002f4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 .....................@
0000300a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00003020 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003036 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
0000304c 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003062 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003078 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
0000308e 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000030a4 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000030ba 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000030d0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000030e6 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AA....................
000030cf 00 00 00 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ....AAAAAAAAAAAAAAAAAA
00003112 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003128 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
0000313e 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003154 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
0000316a 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003180 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003196 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000031ac 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
000031c2 41 41 41 41 41 41 47 43 43 3A 20 28 55 62 75 6E 74 75 20 39 2E 33 AAAAAAGCC: (Ubuntu 9.3
000031d8 2E 30 2D 31 37 75 62 75 6E 74 75 31 7E 32 30 2E 30 34 29 20 39 2E .0-17ubuntu1~20.04) 9.
000031ee 33 2E 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3.0...................
```

Select range from: 12544 to/±length 12743 ✓ Select

Signed 8 bit:	71	Signed 32 bit:	1195590458	Hexadecimal:	47 43 43 3A
Unsigned 8 bit:	71	Unsigned 32 bit:	1195590458	Decimal:	071 067 067 058
Signed 16 bit:	18243	Float 32 bit:	49987.23	Octal:	107 103 103 072
Unsigned 16 bit:	18243	Float 64 bit:	2.00034333882626E+35	Binary:	01000111 01000011 01000011 0C
□ Show little endian decoding		□ Show unsigned as hexadecimal		ASCII Text:	GCC:

First, I split the prefix first, I choose the same size in Task 3, which include 32 bytes "A".



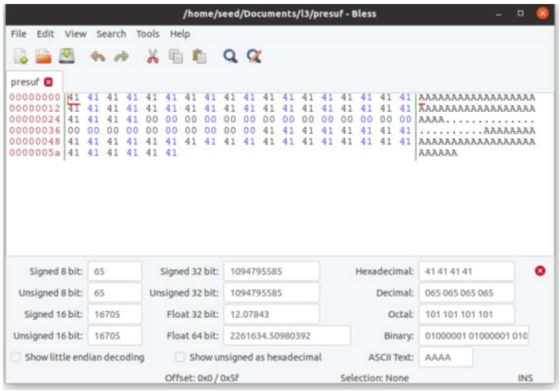
/home/seed/Documents/l3/prefix - Bless

```
00002e68 0A 00 00 00 00 00 00 00 82 00 00 00 00 00 00 00 0B 00 00 00 00 00 ......................
00002e7e 00 00 18 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 ......................
00002e94 00 00 00 00 03 00 00 00 00 00 00 00 B8 3F 00 00 00 00 00 00 02 00 .............?........
00002eaa 00 00 00 00 00 00 18 00 00 00 00 00 00 00 14 00 00 00 00 00 00 00 ......................
00002ec0 07 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00 E0 05 00 00 00 00 ......................
00002ed6 00 00 07 00 00 00 00 00 00 00 20 05 00 00 00 00 00 00 08 00 00 00 .......... ...........
00002eec 00 00 00 00 C0 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00 18 00 ......................
00002f02 00 00 00 00 00 00 1E 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 ......................
00002f18 FB FF FF 6F 00 00 00 00 01 00 00 08 00 00 00 00 FE FF FF 6F 00 00 ...o...............o..
00002f2e 00 00 00 05 00 00 00 00 00 00 FF FF FF 6F 00 00 00 00 01 00 00 00 .............o........
00002f44 00 00 00 00 F0 FF FF 6F 00 00 00 00 F2 04 00 00 00 00 00 00 F9 FF .......o..............
00002f5a FF 6F 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .o....................
00002f70 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002f86 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002f9c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002fb2 00 00 00 00 00 00 C8 3D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .......=..............
00002fc8 00 00 00 00 00 00 00 00 30 10 00 00 00 00 00 00 00 00 00 00 00 00 ........0.............
00002fde 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002f4f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 40 .....................@
0000300a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00003020 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAA
00003036 41 41 41 41 41 41 41 41 41 41                                     AAAAAAAAAA
```

Signed 8 bit:	127	Signed 32 bit:	2135247942	Hexadecimal:	7F 45 4C 46
Unsigned 8 bit:	127	Unsigned 32 bit:	2135247942	Decimal:	127 069 076 070
Signed 16 bit:	32581	Float 32 bit:	2.622539E+38	Octal:	177 105 114 106

To make sure X==Y in the first version, array Y should be split into first 32 bytes and last (200-128-32 = 40) Bytes.

Therefore, I first split T4 at (12544+32 = 12576) bytes and save content in t1, and split t1 at (12352+128+1 = 12481) bytes and save the content into presuf.



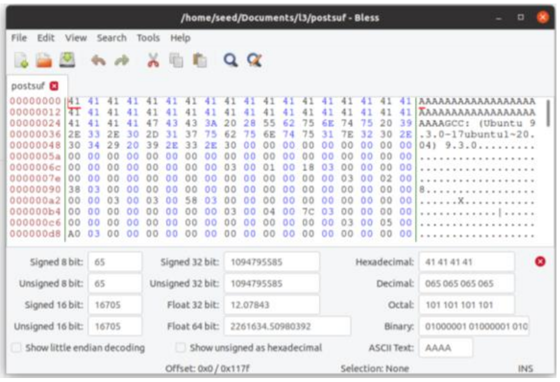
/home/seed/Documents/l3/presuf - Bless

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAA..............
00000036 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 ..........AAAAAAAA
00000048 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
0000005a 41 41 41 41 41 41                                     AAAAAA
```

Signed 8 bit:	65	Signed 32 bit:	1094795585	Hexadecimal:	41 41 41 41
Unsigned 8 bit:	65	Unsigned 32 bit:	1094795585	Decimal:	065 065 065 065
Signed 16 bit:	16705	Float 32 bit:	12.07843	Octal:	101 101 101 101
Unsigned 16 bit:	16705	Float 64 bit:	2261634.50980392	Binary:	01000001 01000001 010
□ Show little endian decoding		□ Show unsigned as hexadecimal		ASCII Text:	AAAA

Offset: 0x0 / 0x5f Selection: None INS

Then, I split T4 at (12576+128+1=12705)bytes and save the content in postsuf.



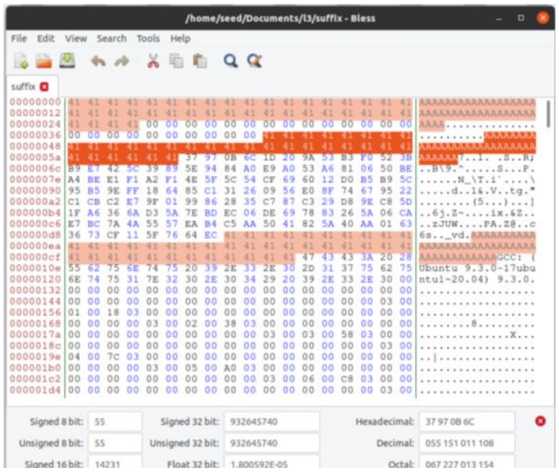
/home/seed/Documents/l3/postsuf - Bless
File Edit View Search Tools Help
postsuf

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 47 43 43 3A 20 28 55 62 75 6E 74 75 20 39 AAAAGCC: (Ubuntu 9
00000036 2E 33 2E 30 2D 31 37 75 62 75 6E 74 75 31 7E 32 30 2E .3.0-17ubuntu1~20.
00000048 30 34 29 20 39 2E 33 2E 30 00 00 00 00 00 00 00 00 00 04) 9.3.0.........
0000005a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000006c 00 00 00 00 00 00 00 00 03 00 01 00 18 03 00 00 00 00 ..................
0000007e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 02 00 ..................
00000090 38 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 8.................
000000a2 00 00 03 00 03 00 58 03 00 00 00 00 00 00 00 00 00 00 ......X...........
000000b4 00 00 00 00 00 00 00 00 03 00 04 00 7C 03 00 00 00 00 ............|.....
000000c6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 05 00 ..................
000000d8 A0 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
```

```
Signed 8 bit: 65        Signed 32 bit: 1094795585     Hexadecimal: 41 41 41 41
Unsigned 8 bit: 65      Unsigned 32 bit: 1094795585   Decimal: 065 065 065 065
Signed 16 bit: 16705    Float 32 bit: 12.07843        Octal: 101 101 101 101
Unsigned 16 bit: 16705  Float 64 bit: 2261634.50980392  Binary: 01000001 01000001 010
□ Show little endian decoding   □ Show unsigned as hexadecimal   ASCII Text: AAAA
Offset: 0x0 / 0x117f    Selection: None    INS
```

The next step is finding P and Q. I use prefix to get two new file “prefix1” and “prefix2” who have the same hash value. Based on the Task1, P and Q should be the last 128 bytes of “prefix1” and “prefix2”, because the prefix is the multiply of 64 bytes. Split last 128 bytes into “prefix1” and “prefix2” and get P and Q.

Then, I cat presuf, P, postsuf to get suffix.



/home/seed/Documents/l3/suffix - Bless
File Edit View Search Tools Help
suffix

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00000024 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAA..............
00000036 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 ..........AAAAAAAA
00000048 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
0000005a 41 41 41 41 41 41 37 97 0B 6C 1D 20 9A 53 B3 F0 52 3B AAAAAA7..l. .S..R;
0000006c B9 E7 42 5C 39 89 5E 94 84 A0 E9 A0 53 A6 81 06 50 BE ..B\9.^.....S...P.
0000007e A4 BE E1 F1 A2 F1 4E 5F 5C 54 CF 69 60 12 D0 B5 B9 5C ......N_\T.i`....\
00000090 95 B5 9E FF 18 64 85 C1 31 26 09 56 E0 8F 74 67 95 22 .....d..1&.V..tg."
000000a2 C1 CB C2 E7 9F 01 99 86 28 35 C7 87 C3 29 D8 9E C8 5D ........(5...)...]
000000b4 1F A6 36 6A D3 5A 7E BD EC 06 DE 69 78 83 26 5A 06 CA ..6j.Z~....ix.&Z..
000000c6 E7 BC 7A 4A 55 57 EA B4 C5 AA 50 41 82 5A 40 AA 01 63 ..zJUW....PA.Z@..c
000000d8 36 73 CF 11 5F 76 64 EC 41 41 41 41 41 41 41 41 41 41 6s.._vd.AAAAAAAAAA
000000ea 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
000000cf 41 41 41 41 41 41 41 41 41 41 41 41 47 43 43 3A 20 28 AAAAAAAAAAAAGCC: (
0000010e 55 62 75 6E 74 75 20 39 2E 33 2E 30 2D 31 37 75 62 75 Ubuntu 9.3.0-17ubu
00000120 6E 74 75 31 7E 32 30 2E 30 34 29 20 39 2E 33 2E 30 00 ntu1~20.04) 9.3.0.
00000132 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000144 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 ..................
00000156 01 00 18 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000168 00 00 00 00 03 00 02 00 38 03 00 00 00 00 00 00 00 00 ........8.........
0000017a 00 00 00 00 00 00 00 00 00 00 03 00 03 00 58 03 00 00 ..............X...
0000018c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 ..................
0000019e 04 00 7C 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..|...............
000001b0 00 00 00 00 03 00 05 00 A0 03 00 00 00 00 00 00 00 00 ..................
000001c2 00 00 00 00 00 00 00 00 00 00 03 00 06 00 C8 03 00 00 ..................
000001d4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 ..................
```

```
Signed 8 bit: 55        Signed 32 bit: 932645740      Hexadecimal: 37 97 0B 6C
Unsigned 8 bit: 55      Unsigned 32 bit: 932645740    Decimal: 055 151 011 108
Signed 16 bit: 14231    Float 32 bit: 1.800592E-05    Octal: 067 227 013 154
```

Then, cat prefix1, suffix to get t4out1; cat prefix2, suffix to get t4out2.

Finally, verify t4out1, t4out2. The hash value of t4out1 and t4out2 are same, and t4out1 and t4out2 can be executable. However, their outputs are not same. T4out1 output that X and Y are same, and t4out2 output that X and Y are different.