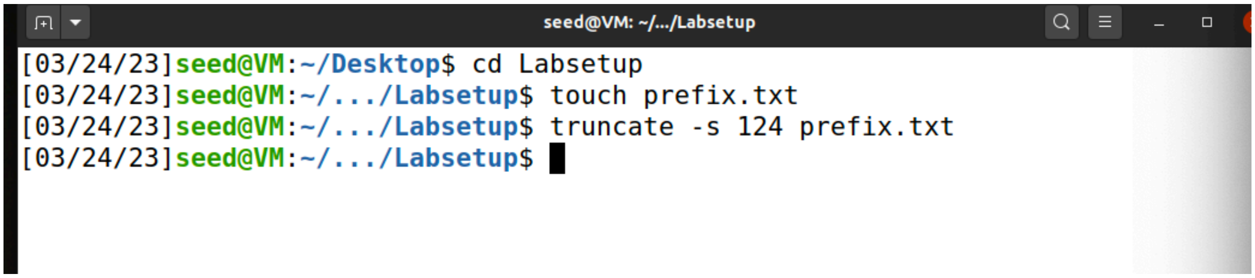
Christian Kuhn
Lab 2
Friday, March 24th

Task 1

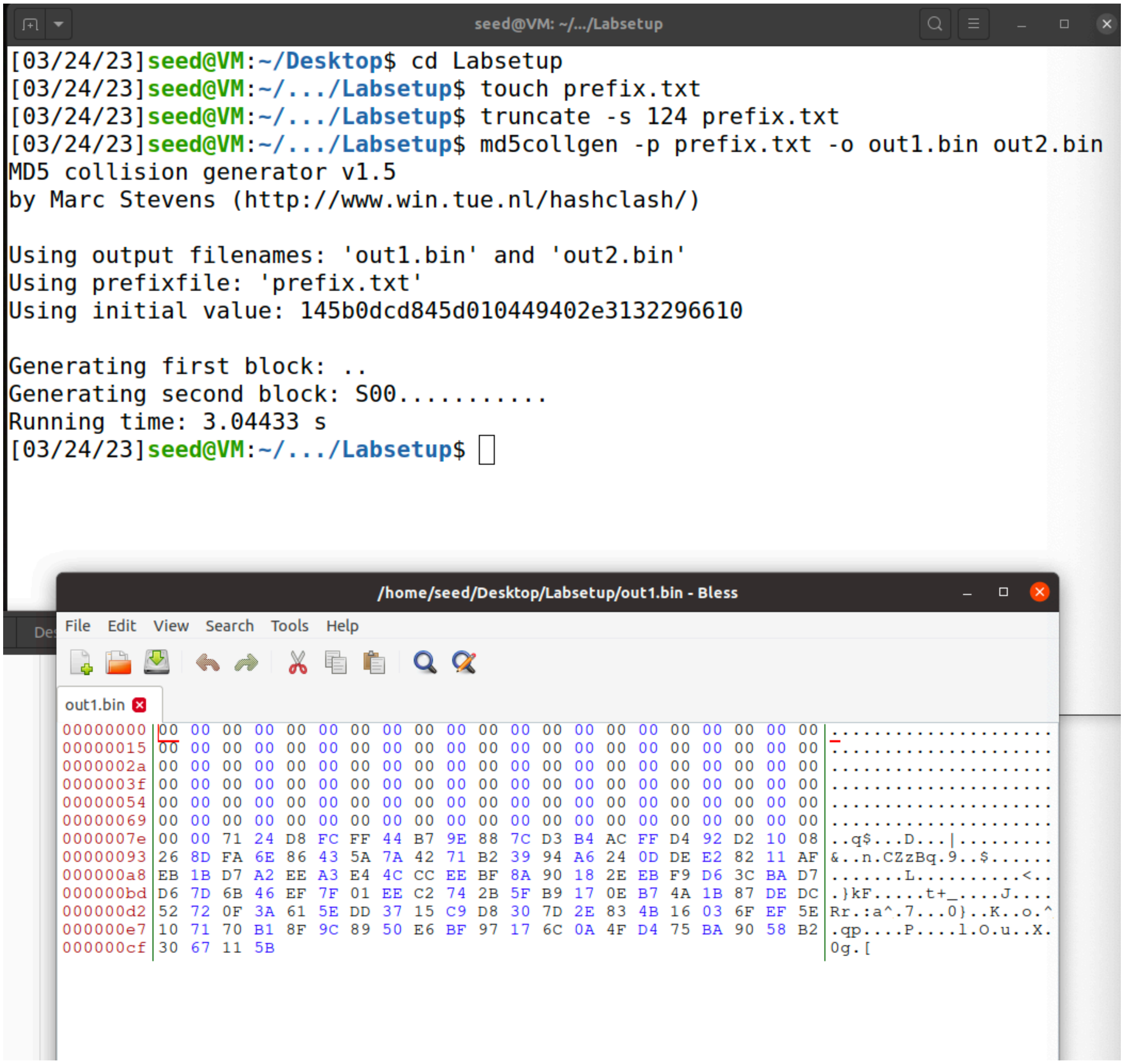
Question 1:

To test out this function I created a file prefix.txt and truncated it using the following command



```
[03/24/23]seed@VM:~/Desktop$ cd Labsetup
[03/24/23]seed@VM:~/.../Labsetup$ touch prefix.txt
[03/24/23]seed@VM:~/.../Labsetup$ truncate -s 124 prefix.txt
[03/24/23]seed@VM:~/.../Labsetup$ 
```

I then ran the md5collgen command and we can see the output files in bless hex editor:



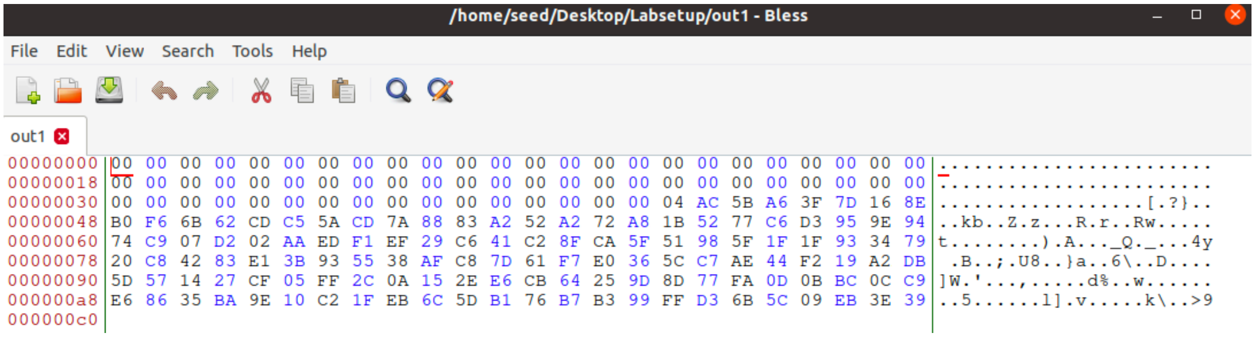
```
[03/24/23]seed@VM:~/Desktop$ cd Labsetup
[03/24/23]seed@VM:~/.../Labsetup$ touch prefix.txt
[03/24/23]seed@VM:~/.../Labsetup$ truncate -s 124 prefix.txt
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 145b0dcd845d010449402e3132296610

Generating first block: ..
Generating second block: S00...........
Running time: 3.04433 s
[03/24/23]seed@VM:~/.../Labsetup$ 
```

```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
00000015 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
0000002a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
0000003f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
00000054 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
00000069 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....................
0000007e 00 00 71 24 D8 FC FF 44 B7 9E 88 7C D3 B4 AC FF D4 92 D2 10 08 ..q$...D...|.........
00000093 26 8D FA 6E 86 43 5A 7A 42 71 B2 39 94 A6 24 0D DE E2 82 11 AF &..n.CZzBq.9..$......
000000a8 EB 1B D7 A2 EE A3 E4 4C CC EE BF 8A 90 18 2E EB F9 D6 3C BA D7 .......L.........<..
000000bd D6 7D 6B 46 EF 7F 01 EE C2 74 2B 5F B9 17 0E B7 4A 1B 87 DE DC .}kF.....t+_....J....
000000d2 52 72 0F 3A 61 5E DD 37 15 C9 D8 30 7D 2E 83 4B 16 03 6F EF 5E Rr.:a^.7...0}..K..o.^
000000e7 10 71 70 B1 8F 9C 89 50 E6 BF 97 17 6C 0A 4F D4 75 BA 90 58 B2 .qp....P....l.O.u..X.
000000cf 30 67 11 5B                                                    0g.[
```

We can observe that padding has been added with zeroes if our prefix file is not a multiple of 64.

Question 2:

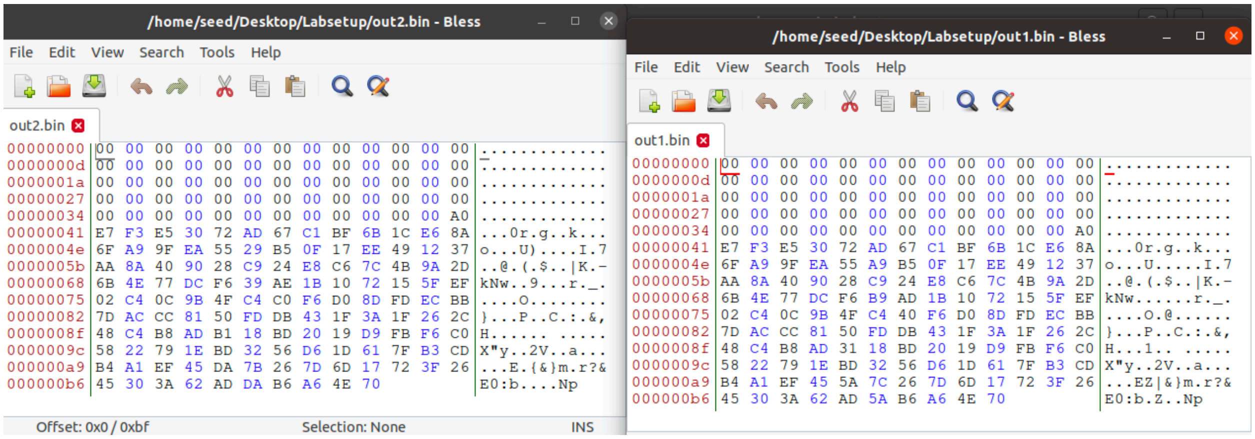
I created a prefix file with exactly 64 bytes using the same process as above. This is our result in the output file:


/home/seed/Desktop/Labsetup/out1 - Bless
File Edit View Search Tools Help
out1

```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........................
00000018 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 AC 5B A6 3F 7D 16 8E ..................[.?}..
00000048 B0 F6 6B 62 CD C5 5A CD 7A 88 83 A2 52 A2 72 A8 1B 52 77 C6 D3 95 9E 94 ..kb..Z.z...R.r..Rw.....
00000060 74 C9 07 D2 02 AA ED F1 EF 29 C6 41 C2 8F CA 5F 51 98 5F 1F 1F 93 34 79 t........).A..._Q._...4y
00000078 20 C8 42 83 E1 3B 93 55 38 AF C8 7D 61 F7 E0 36 5C C7 AE 44 F2 19 A2 DB  .B..;.U8..}a..6\..D....
00000090 5D 57 14 27 CF 05 FF 2C 0A 15 2E E6 CB 64 25 9D 8D 77 FA 0D 0B BC 0C C9 ]W.'...,.....d%..w......
000000a8 E6 86 35 BA 9E 10 C2 1F EB 6C 5D B1 76 B7 B3 99 FF D3 6B 5C 09 EB 3E 39 ..5......l].v.....k\..>9
000000c0
```

No padding is observed.

Question 3:


/home/seed/Desktop/Labsetup/out2.bin - Bless
File Edit View Search Tools Help
out2.bin

```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
0000000d 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
0000001a 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
00000027 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
00000034 00 00 00 00 00 00 00 00 00 00 00 00 A0 .............
00000041 E7 F3 E5 30 72 AD 67 C1 BF 6B 1C E6 8A ...0r.g..k...
0000004e 6F A9 9F EA 55 29 B5 0F 17 EE 49 12 37 o...U)....I.7
0000005b AA 8A 40 90 28 C9 24 E8 C6 7C 4B 9A 2D ..@.(.$..|K.-
00000068 6B 4E 77 DC F6 39 AE 1B 10 72 15 5F EF kNw..9...r._.
00000075 02 C4 0C 9B 4F C4 C0 F6 D0 8D FD EC BB ....O........
00000082 7D AC CC 81 50 FD DB 43 1F 3A 1F 26 2C }...P..C.:.&,
0000008f 48 C4 B8 AD B1 18 BD 20 19 D9 FB F6 C0 H...... .....
0000009c 58 22 79 1E BD 32 56 D6 1D 61 7F B3 CD X"y..2V..a...
000000a9 B4 A1 EF 45 DA 7B 26 7D 6D 17 72 3F 26 ...E.{&}m.r?&
000000b6 45 30 3A 62 AD DA B6 A6 4E 70          E0:b....Np
```

Offset: 0x0 / 0xbf   Selection: None   INS

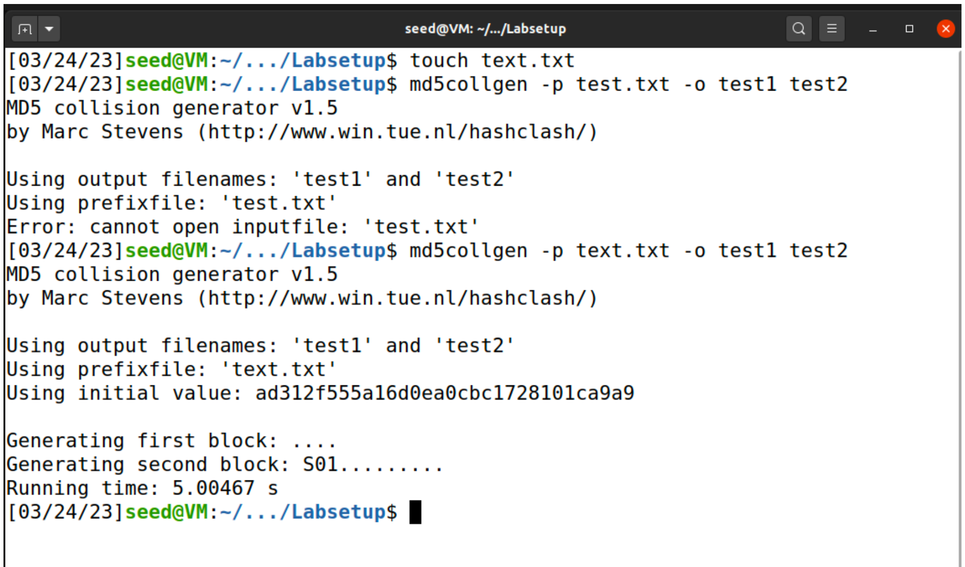
/home/seed/Desktop/Labsetup/out1.bin - Bless
File Edit View Search Tools Help
out1.bin

```
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
0000000d 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
0000001a 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
00000027 00 00 00 00 00 00 00 00 00 00 00 00 00 .............
00000034 00 00 00 00 00 00 00 00 00 00 00 00 A0 .............
00000041 E7 F3 E5 30 72 AD 67 C1 BF 6B 1C E6 8A ...0r.g..k...
0000004e 6F A9 9F EA 55 A9 B5 0F 17 EE 49 12 37 o...U.....I.7
0000005b AA 8A 40 90 28 C9 24 E8 C6 7C 4B 9A 2D ..@.(.$..|K.-
00000068 6B 4E 77 DC F6 B9 AD 1B 10 72 15 5F EF kNw......r._.
00000075 02 C4 0C 9B 4F C4 40 F6 D0 8D FD EC BB ....O.@......
00000082 7D AC CC 81 50 FD DB 43 1F 3A 1F 26 2C }...P..C.:.&,
0000008f 48 C4 B8 AD 31 18 BD 20 19 D9 FB F6 C0 H...1.. .....
0000009c 58 22 79 1E BD 32 56 D6 1D 61 7F B3 CD X"y..2V..a...
000000a9 B4 A1 EF 45 5A 7C 26 7D 6D 17 72 3F 26 ...EZ|&}m.r?&
000000b6 45 30 3A 62 AD 5A B6 A6 4E 70          E0:b.Z..Np
```

Not all of the bytes are different, however if you follow along you will not that there are a few differences.

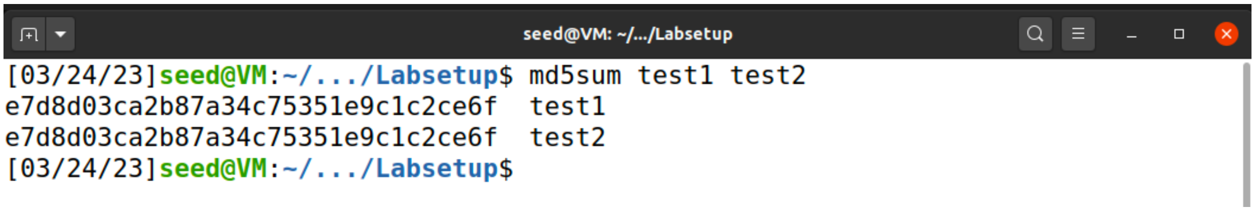
Task 2:

To test the following we will create test.txt and run the following:



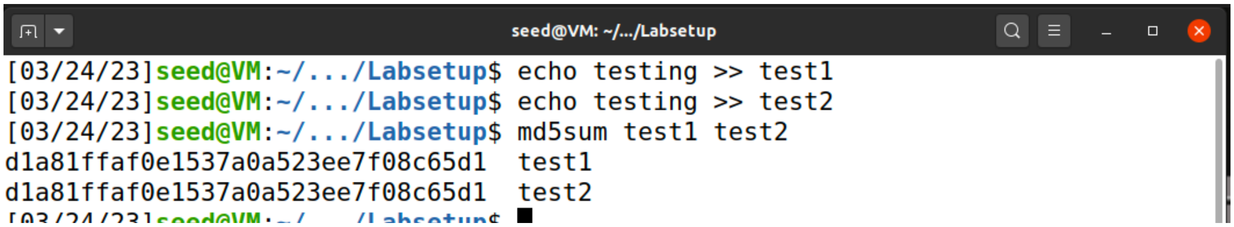
```
[03/24/23]seed@VM:~/.../Labsetup$ touch text.txt
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p test.txt -o test1 test2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'test1' and 'test2'
Using prefixfile: 'test.txt'
Error: cannot open inputfile: 'test.txt'
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p text.txt -o test1 test2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'test1' and 'test2'
Using prefixfile: 'text.txt'
Using initial value: ad312f555a16d0ea0cbc1728101ca9a9

Generating first block: ....
Generating second block: S01.........
Running time: 5.00467 s
[03/24/23]seed@VM:~/.../Labsetup$ 
```

We can now verify that the md5 hashes are the same using the following:



```
[03/24/23]seed@VM:~/.../Labsetup$ md5sum test1 test2
e7d8d03ca2b87a34c75351e9c1c2ce6f  test1
e7d8d03ca2b87a34c75351e9c1c2ce6f  test2
[03/24/23]seed@VM:~/.../Labsetup$
```

Now we will append a string to the end of both files and see what changes we see in the md5 hash:

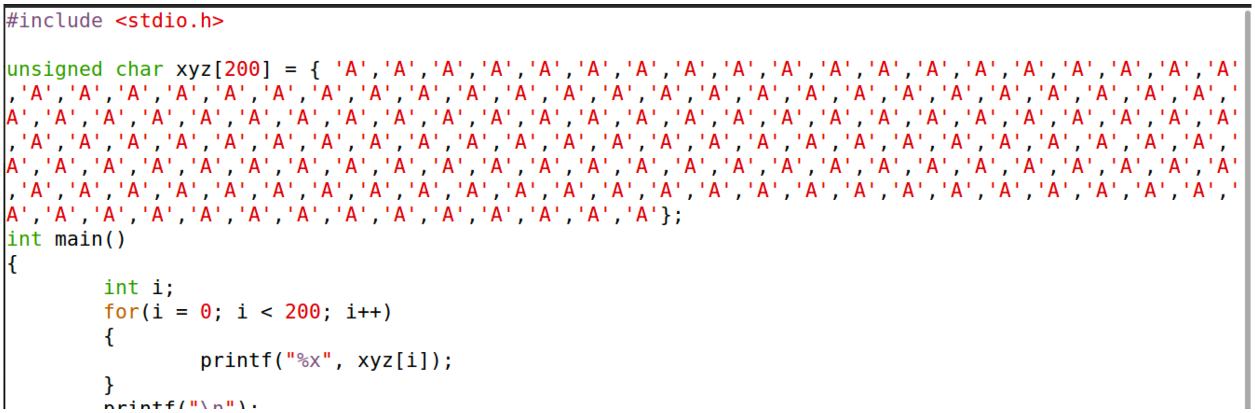


```
[03/24/23]seed@VM:~/.../Labsetup$ echo testing >> test1
[03/24/23]seed@VM:~/.../Labsetup$ echo testing >> test2
[03/24/23]seed@VM:~/.../Labsetup$ md5sum test1 test2
d1a81ffaf0e1537a0a523ee7f08c65d1  test1
d1a81ffaf0e1537a0a523ee7f08c65d1  test2
[03/24/23]seed@VM:~/.../Labsetup$ 
```

As we can see the md5 hashes remained the same even after appending a string

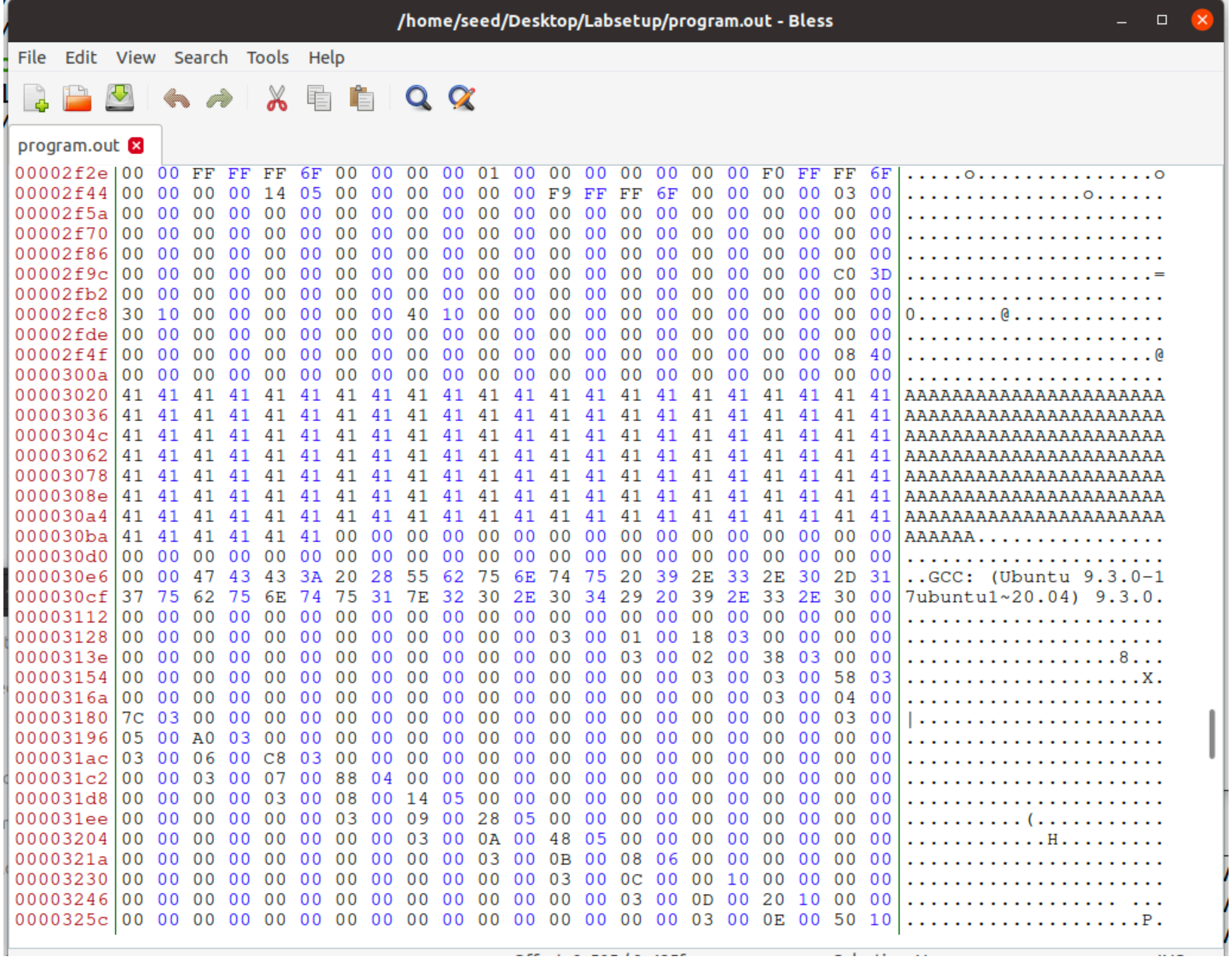
Task 3:

Here is the c program we will be using:



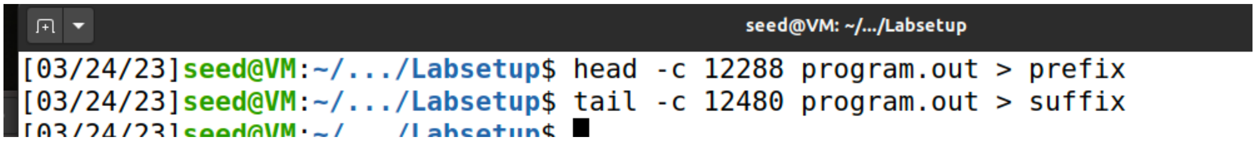
```
#include <stdio.h>

unsigned char xyz[200] = { 'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
A','A','A','A','A','A','A','A','A','A','A','A','A','A'};
int main()
{
        int i;
        for(i = 0; i < 200; i++)
        {
                printf("%x", xyz[i]);
        }
        printf("\n");
```

We can compile the above program using the following command “gcc program.c -o program.out” and then see the following from output. You can clearly see the prefix, 128-byte region, and a suffix.


/home/seed/Desktop/Labsetup/program.out - Bless
File Edit View Search Tools Help
program.out

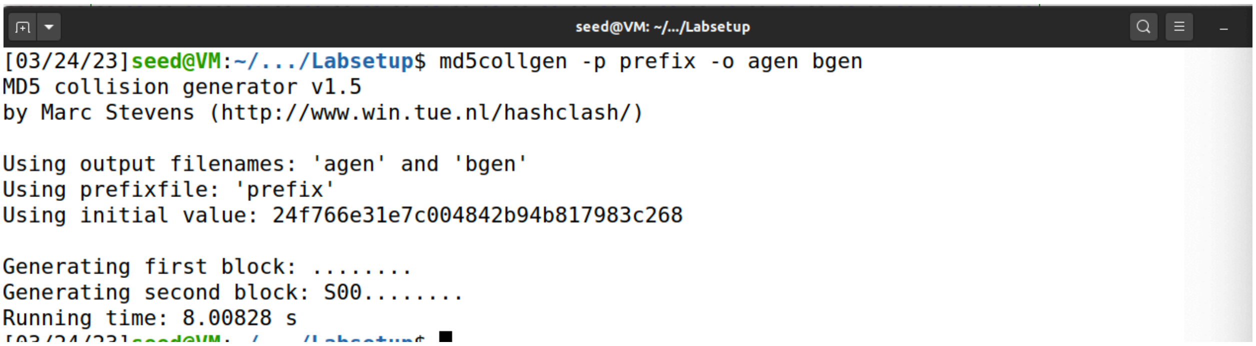
```
00002f2e 00 00 FF FF FF 6F 00 00 00 00 01 00 00 00 00 00 00 00 F0 FF FF 6F .....o...............o
00002f44 00 00 00 00 14 05 00 00 00 00 00 00 F9 FF FF 6F 00 00 00 00 03 00 ................o.....
00002f5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002f70 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 C0 3D .....................=
00002fb2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00002fc8 30 10 00 00 00 00 00 00 40 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 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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAAAA................
000030d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000030e6 00 00 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 30 00 7ubuntu1~20.04) 9.3.0.
00003112 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
00003128 00 00 00 00 00 00 00 00 00 00 00 00 03 00 01 00 18 03 00 00 00 00 ......................
0000313e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 02 00 38 03 00 00 ...................8..
00003154 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 03 00 58 03 ....................X.
0000316a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 04 00 ......................
00003180 7C 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 |.....................
00003196 05 00 A0 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000031ac 03 00 06 00 C8 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000031c2 00 00 03 00 07 00 88 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000031d8 00 00 00 00 03 00 08 00 14 05 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000031ee 00 00 00 00 00 00 03 00 09 00 28 05 00 00 00 00 00 00 00 00 00 00 ..........(...........
00003204 00 00 00 00 00 00 00 00 03 00 0A 00 48 05 00 00 00 00 00 00 00 00 ............H.........
0000321a 00 00 00 00 00 00 00 00 00 00 03 00 0B 00 08 06 00 00 00 00 00 00 ......................
00003230 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0C 00 00 10 00 00 00 00 ......................
00003246 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0D 00 20 10 00 00 ...................  ...
0000325c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0E 00 50 10 ....................P.
```

To divide our output up we run the following:



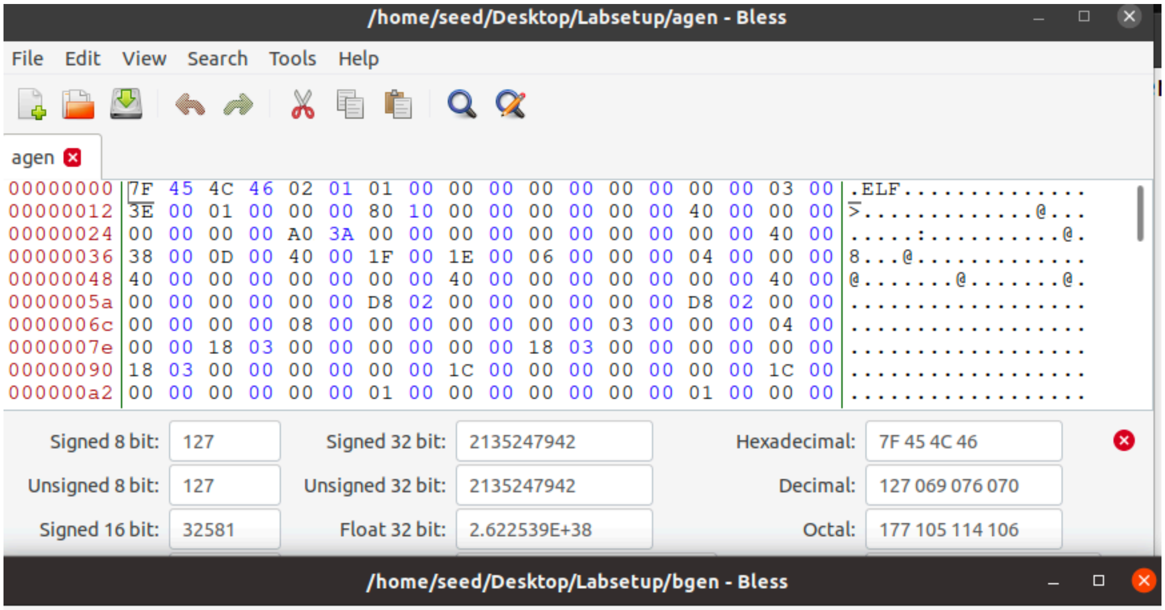
```
[03/24/23]seed@VM:~/.../Labsetup$ head -c 12288 program.out > prefix
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 12480 program.out > suffix
[03/24/23]seed@VM:~/.../Labsetup$
```

We now run the following:



```
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p prefix -o agen bgen
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'agen' and 'bgen'
Using prefixfile: 'prefix'
Using initial value: 24f766e31e7c004842b94b817983c268

Generating first block: ........
Generating second block: S00........
Running time: 8.00828 s
[03/24/23]seed@VM:~/.../Labsetup$
```

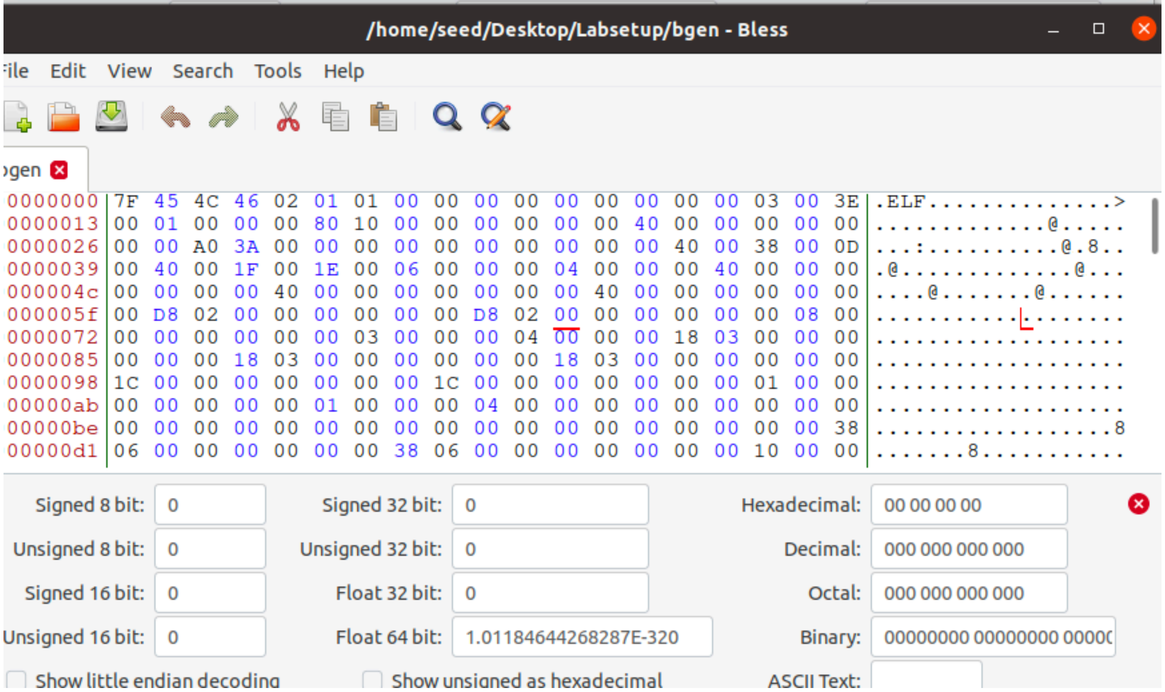
We can now compare the ages and bgen files in bless:



/home/seed/Desktop/Labsetup/agen - Bless
File Edit View Search Tools Help
agen

```
00000000 7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00 03 00 .ELF..............
00000012 3E 00 01 00 00 00 80 10 00 00 00 00 00 00 40 00 00 00 >.............@...
00000024 00 00 00 00 A0 3A 00 00 00 00 00 00 00 00 00 00 40 00 .....:..........@.
00000036 38 00 0D 00 40 00 1F 00 1E 00 06 00 00 00 04 00 00 00 8...@.............
00000048 40 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 40 00 @.......@.......@.
0000005a 00 00 00 00 00 00 D8 02 00 00 00 00 00 00 D8 02 00 00 ..................
0000006c 00 00 00 00 08 00 00 00 00 00 00 00 03 00 00 00 04 00 ..................
0000007e 00 00 18 03 00 00 00 00 00 00 18 03 00 00 00 00 00 00 ..................
00000090 18 03 00 00 00 00 00 00 1C 00 00 00 00 00 00 00 1C 00 ..................
000000a2 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 ..................
```

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

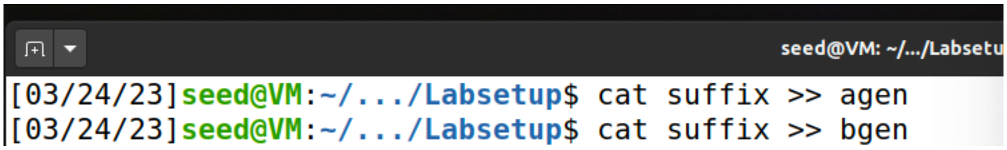


/home/seed/Desktop/Labsetup/bgen - Bless
File Edit View Search Tools Help
bgen

```
0000000 7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00 03 00 3E .ELF..............>
0000013 00 01 00 00 00 80 10 00 00 00 00 00 00 40 00 00 00 00 00 .............@.....
0000026 00 00 A0 3A 00 00 00 00 00 00 00 00 00 00 40 00 38 00 0D ...:..........@.8..
0000039 00 40 00 1F 00 1E 00 06 00 00 00 04 00 00 00 40 00 00 00 .@.............@...
000004c 00 00 00 00 40 00 00 00 00 00 00 00 40 00 00 00 00 00 00 ....@.......@......
000005f 00 D8 02 00 00 00 00 00 00 D8 02 00 00 00 00 00 00 08 00 ...................
0000072 00 00 00 00 00 00 03 00 00 00 04 00 00 00 18 03 00 00 00 ...................
0000085 00 00 00 18 03 00 00 00 00 00 00 18 03 00 00 00 00 00 00 ...................
0000098 1C 00 00 00 00 00 00 00 1C 00 00 00 00 00 00 00 01 00 00 ...................
00000ab 00 00 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00 00 00 ...................
00000be 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 38 ..................8
00000d1 06 00 00 00 00 00 00 38 06 00 00 00 00 00 00 00 10 00 00 .......8...........
```

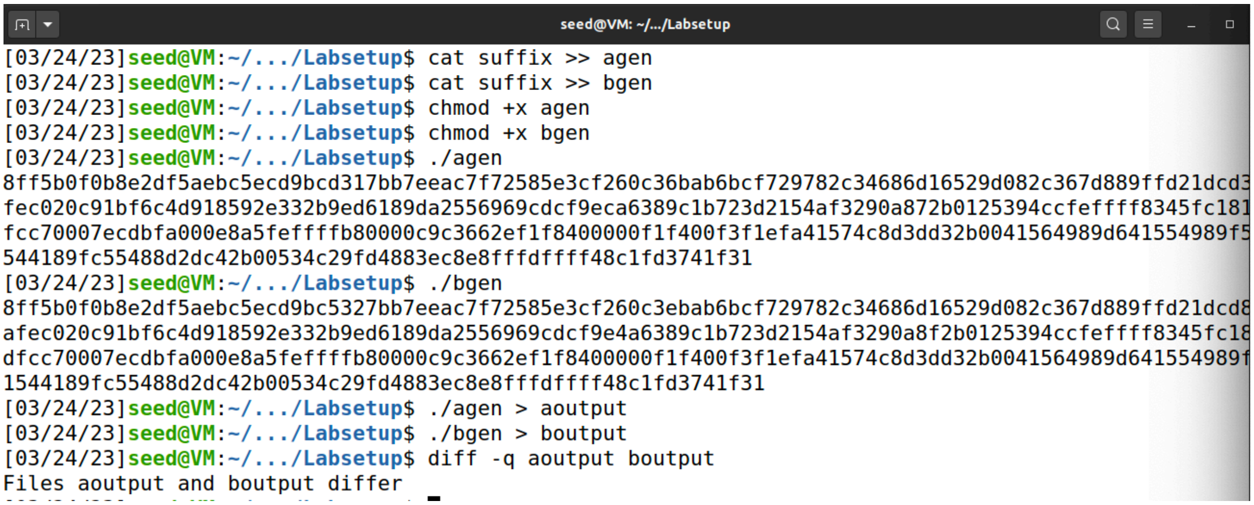
Signed 8 bit: 0
Unsigned 8 bit: 0
Signed 16 bit: 0
Unsigned 16 bit: 0
Signed 32 bit: 0
Unsigned 32 bit: 0
Float 32 bit: 0
Float 64 bit: 1.01184644268287E-320
Hexadecimal: 00 00 00 00
Decimal: 000 000 000 000
Octal: 000 000 000 000
Binary: 00000000 00000000 0000(
ASCII Text:
□ Show little endian decoding
□ Show unsigned as hexadecimal

We now have two files with the same md5 hash, but two different suffixes. We can append our earlier suffix to the end by running:



```
[03/24/23]seed@VM:~/.../Labsetup$ cat suffix >> agen
[03/24/23]seed@VM:~/.../Labsetup$ cat suffix >> bgen
```

We can now run both files and lets see if there is a difference:

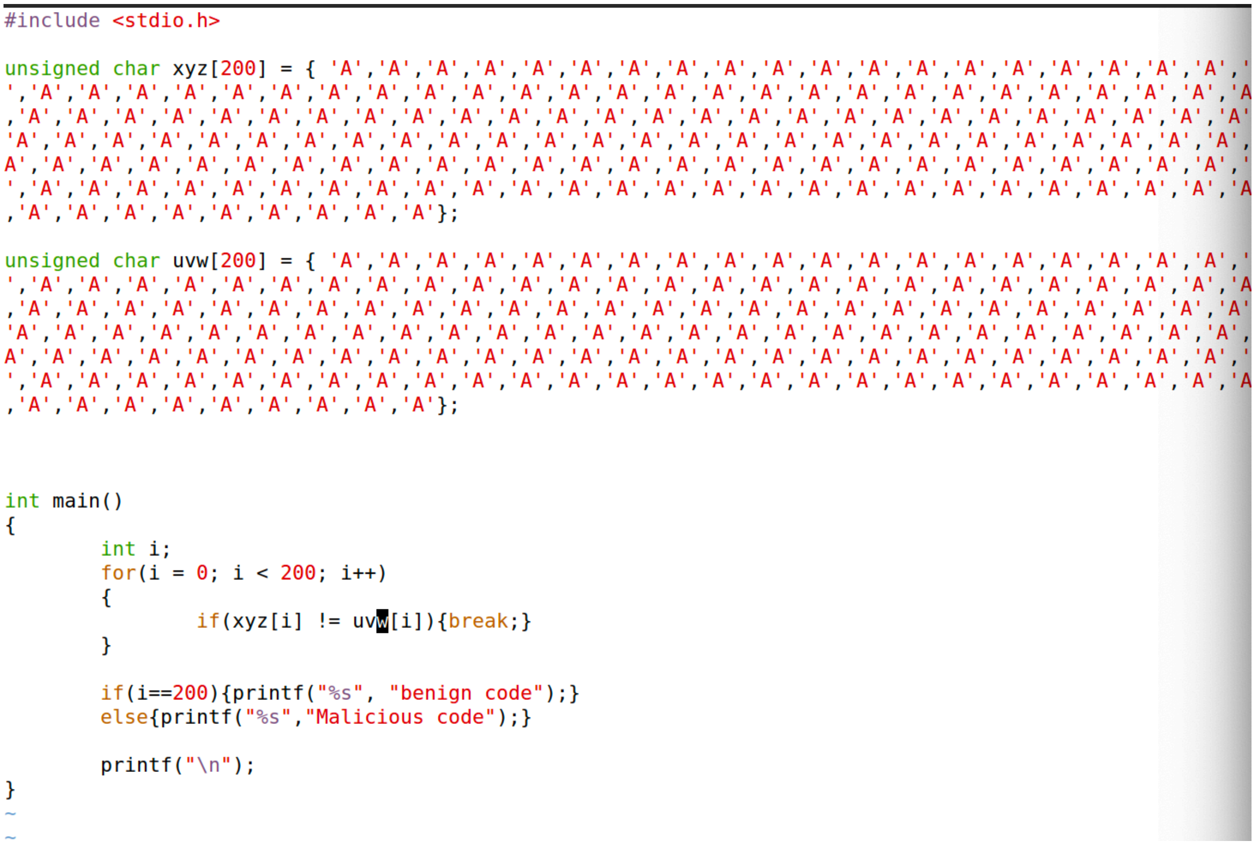


```
[03/24/23]seed@VM:~/.../Labsetup$ cat suffix >> agen
[03/24/23]seed@VM:~/.../Labsetup$ cat suffix >> bgen
[03/24/23]seed@VM:~/.../Labsetup$ chmod +x agen
[03/24/23]seed@VM:~/.../Labsetup$ chmod +x bgen
[03/24/23]seed@VM:~/.../Labsetup$ ./agen
8ff5b0f0b8e2df5aebc5ecd9bcd317bb7eeac7f72585e3cf260c36bab6bcf729782c34686d16529d082c367d889ffd21dcd3
fec020c91bf6c4d918592e332b9ed6189da2556969cdcf9eca6389c1b723d2154af3290a872b0125394ccfeffff8345fc181
fcc70007ecdbfa000e8a5feffffb80000c9c3662ef1f8400000f1f400f3f1efa41574c8d3dd32b0041564989d641554989f5
544189fc55488d2dc42b00534c29fd4883ec8e8fffdffff48c1fd3741f31
[03/24/23]seed@VM:~/.../Labsetup$ ./bgen
8ff5b0f0b8e2df5aebc5ecd9bc5327bb7eeac7f72585e3cf260c3ebab6bcf729782c34686d16529d082c367d889ffd21dcd8
afec020c91bf6c4d918592e332b9ed6189da2556969cdcf9e4a6389c1b723d2154af3290a8f2b0125394ccfeffff8345fc18
dfcc70007ecdbfa000e8a5feffffb80000c9c3662ef1f8400000f1f400f3f1efa41574c8d3dd32b0041564989d641554989f
1544189fc55488d2dc42b00534c29fd4883ec8e8fffdffff48c1fd3741f31
[03/24/23]seed@VM:~/.../Labsetup$ ./agen > aoutput
[03/24/23]seed@VM:~/.../Labsetup$ ./bgen > boutput
[03/24/23]seed@VM:~/.../Labsetup$ diff -q aoutput boutput
Files aoutput and boutput differ
```

We can see that there is actually a difference between the two outputs.

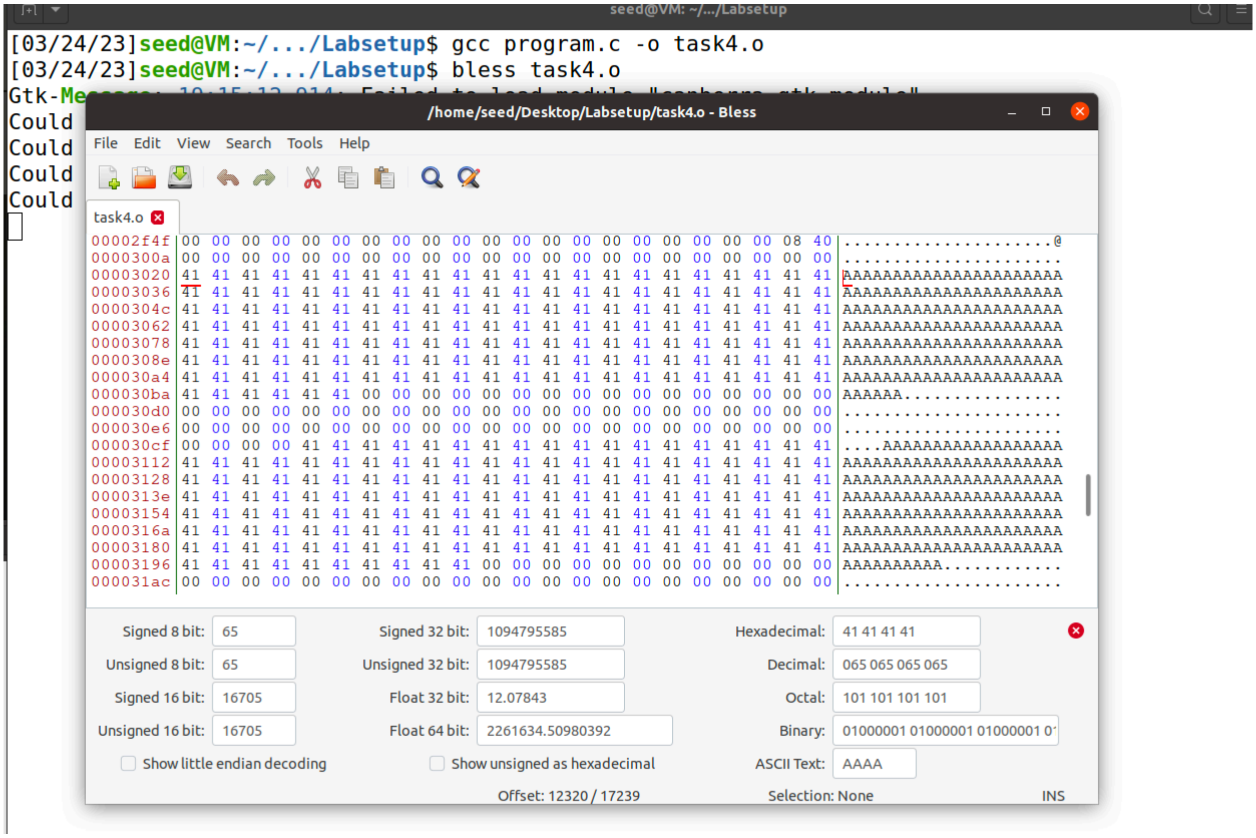
Task 4:

New Program:



```
#include <stdio.h>

unsigned char xyz[200] = { 'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A',
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A'};

unsigned char uvw[200] = { 'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A',
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','
','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A
,'A','A','A','A','A','A','A','A','A'};



int main()
{
        int i;
        for(i = 0; i < 200; i++)
        {
                if(xyz[i] != uvw[i]){break;}
        }

        if(i==200){printf("%s", "benign code");}
        else{printf("%s","Malicious code");}

        printf("\n");
}
~
~
```

Here is our out file in hex editor after compiling the program:



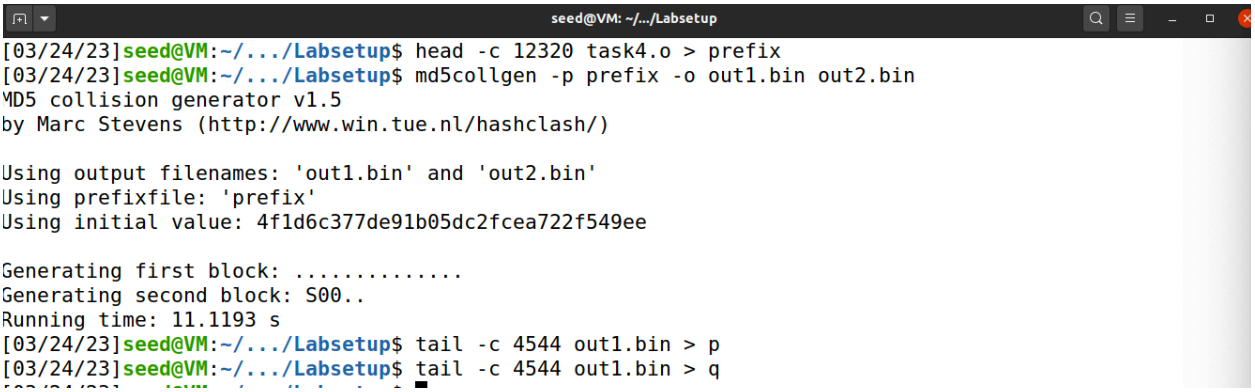
```
[03/24/23]seed@VM:~/.../Labsetup$ gcc program.c -o task4.o
[03/24/23]seed@VM:~/.../Labsetup$ bless task4.o
Gtk-Message: 10:15:12.914: Failed to load module "canberra-gtk-module"
Could
Could
Could
Could
```

```
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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAAAA................
000030d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
000030e6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
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 00 00 00 00 00 00 00 00 00 00 00 00 AAAAAAAAAA............
000031ac 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ......................
```

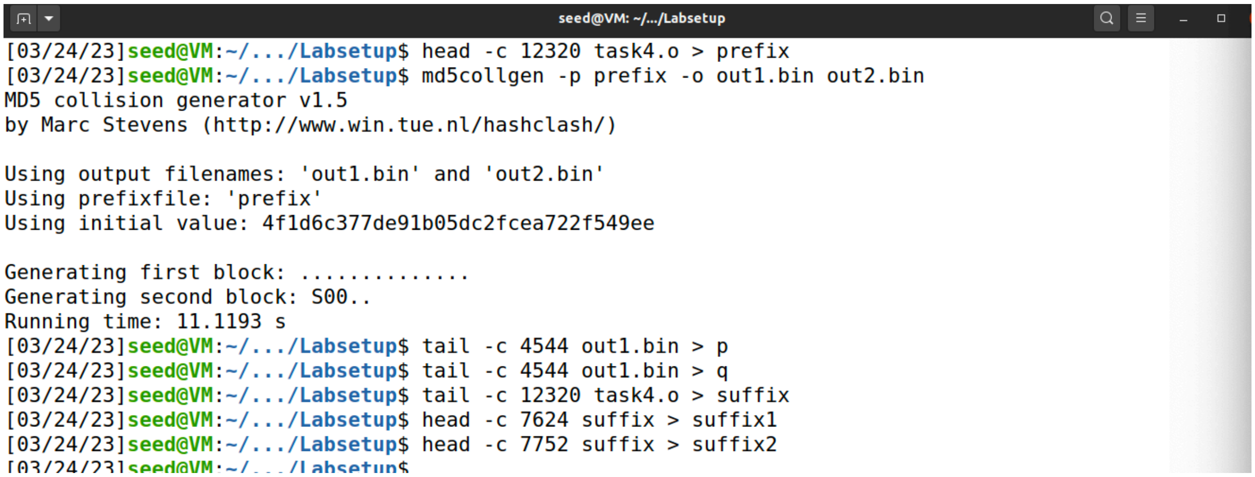
Signed 8 bit:	65	Signed 32 bit:	1094795585	Hexadecimal:	41 41 41 41
Unsigned 8 bit:	65	Unsigned 32 bit:	1094795585	Decimal:	065 065 065 065
Signed 16 bit:	16705	Float 32 bit:	12.07843	Octal:	101 101 101 101
Unsigned 16 bit:	16705	Float 64 bit:	2261634.50980392	Binary:	01000001 01000001 01000001 0
□ Show little endian decoding		□ Show unsigned as hexadecimal		ASCII Text:	AAAA

Offset: 12320 / 17239 Selection: None INS

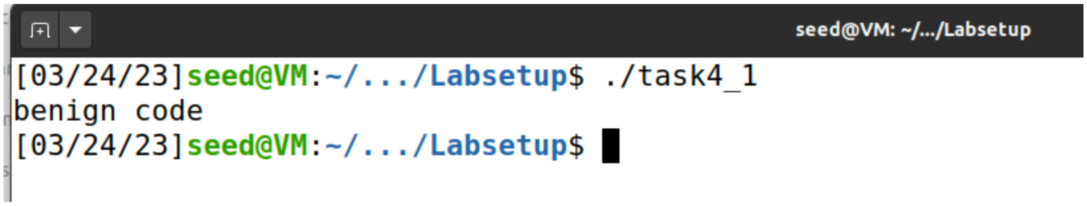
Lets now isolate our bits:



```
[03/24/23]seed@VM:~/.../Labsetup$ head -c 12320 task4.o > prefix
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p prefix -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'
Using initial value: 4f1d6c377de91b05dc2fcea722f549ee

Generating first block: ..............
Generating second block: S00..
Running time: 11.1193 s
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 4544 out1.bin > p
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 4544 out1.bin > q
[03/24/23]seed@VM:~/.../Labsetup$
```



```
[03/24/23]seed@VM:~/.../Labsetup$ head -c 12320 task4.o > prefix
[03/24/23]seed@VM:~/.../Labsetup$ md5collgen -p prefix -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'
Using initial value: 4f1d6c377de91b05dc2fcea722f549ee

Generating first block: ..............
Generating second block: S00..
Running time: 11.1193 s
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 4544 out1.bin > p
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 4544 out1.bin > q
[03/24/23]seed@VM:~/.../Labsetup$ tail -c 12320 task4.o > suffix
[03/24/23]seed@VM:~/.../Labsetup$ head -c 7624 suffix > suffix1
[03/24/23]seed@VM:~/.../Labsetup$ head -c 7752 suffix > suffix2
[03/24/23]seed@VM:~/.../Labsetup$
```



```
[03/24/23]seed@VM:~/.../Labsetup$ ./task4_1
benign code
[03/24/23]seed@VM:~/.../Labsetup$ 
```

Benign code is returned