APPLIED CRYPTOGRAPHY

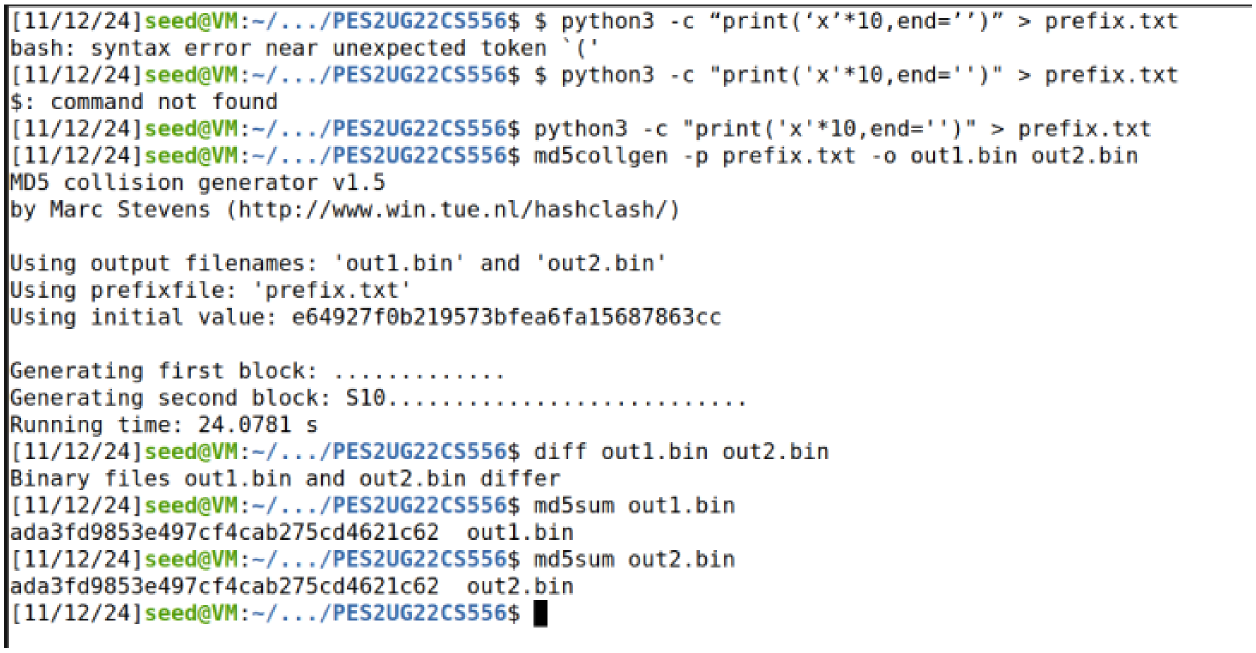
Lab 7: MD5 Collision Attack Lab

Name: Siri N Shetty SRN: PES2UG22CS556

Problem 1: Generating Two Different Files with the Same MD5 Hash

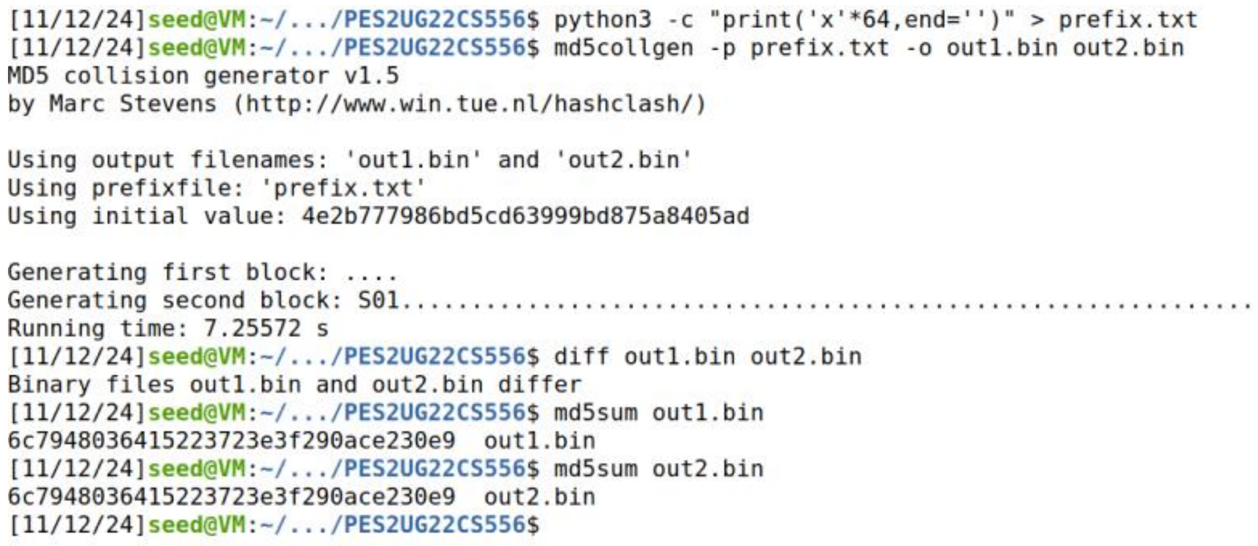
Step 1 : Run the 5 commands for length 10.
Step 2 : Run the 5 commands for length greater than 64.
Step 3 : Run the 5 commands for length 64.

Expected Deliverables -
i) Output Screenshot (Terminal should have SRN visible) for step 1



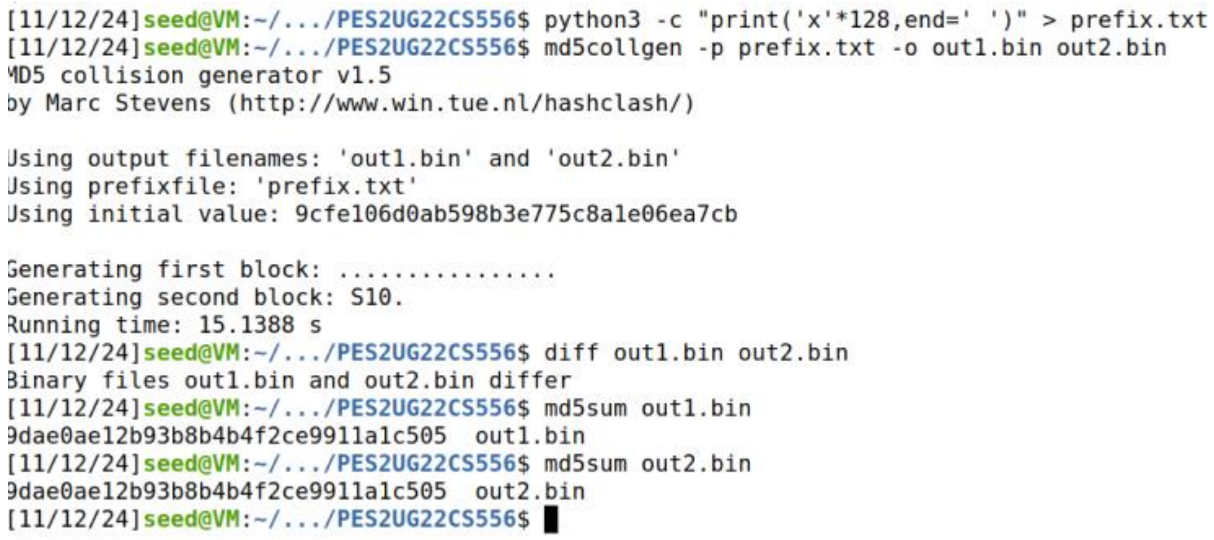
```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ $ python3 -c “print(‘x’*10,end=’’)” > prefix.txt
bash: syntax error near unexpected token `('
[11/12/24]seed@VM:~/.../PES2UG22CS556$ $ python3 -c "print('x'*10,end='')" > prefix.txt
$: command not found
[11/12/24]seed@VM:~/.../PES2UG22CS556$ python3 -c "print('x'*10,end='')" > prefix.txt
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 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: e64927f0b219573bfea6fa15687863cc

Generating first block: .............
Generating second block: S10...........................
Running time: 24.0781 s
[11/12/24]seed@VM:~/.../PES2UG22CS556$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out1.bin
ada3fd9853e497cf4cab275cd4621c62  out1.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out2.bin
ada3fd9853e497cf4cab275cd4621c62  out2.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 
```

ii) Output Screenshot (Terminal should have SRN visible) for step 2



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ python3 -c "print('x'*64,end='')" > prefix.txt
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 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: 4e2b777986bd5cd63999bd875a8405ad

Generating first block: ....
Generating second block: S01...........................................................
Running time: 7.25572 s
[11/12/24]seed@VM:~/.../PES2UG22CS556$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out1.bin
6c7948036415223723e3f290ace230e9  out1.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out2.bin
6c7948036415223723e3f290ace230e9  out2.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$
```

iii) Output Screenshot (Terminal should have SRN visible) for step 3



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ python3 -c "print('x'*128,end=' ')" > prefix.txt
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 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: 9cfe106d0ab598b3e775c8a1e06ea7cb

Generating first block: ...............
Generating second block: S10.
Running time: 15.1388 s
[11/12/24]seed@VM:~/.../PES2UG22CS556$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out1.bin
9dae0ae12b93b8b4b4f2ce9911a1c505  out1.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum out2.bin
9dae0ae12b93b8b4b4f2ce9911a1c505  out2.bin
[11/12/24]seed@VM:~/.../PES2UG22CS556$ █
```

iv) Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different. {answer for all of above i), ii), iii)}

Prefix 1 has 10 bytes: This file is not a multiple of 64 bytes. The MD5 hashing algorithm works on 64-byte blocks, so when the length of prefix is less than 64 bytes, the padding will be added to ensure that the input is a multiple of 64 bytes.

Prefix 2 has 64 bytes: Since this is exactly 64 bytes, the MD5 algorithm can directly process this block without padding.

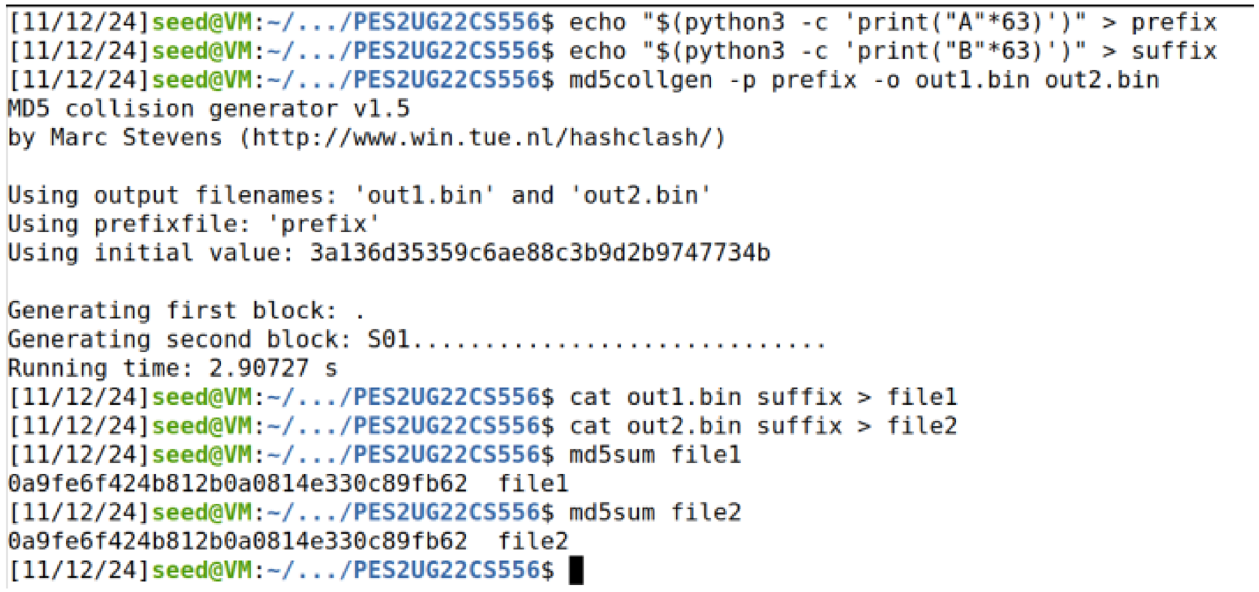
Prefix 3 has 70 bytes: This file exceeds 64 bytes, meaning it spills into the second block. The MD5 algorithm will need to process the first 64-byte block and then the remaining 6 bytes with padding to complete the second block.

No, the 128 bytes generated by md5collgen are not completely different. The tool introduces a controlled change that allows two distinct files to share the same MD5 hash. The differences in the files will likely be localized to certain bytes that md5collgen manipulates to achieve the hash collision.

Problem 2 : Understanding MD5's Property

Step : Run the 7 commands

Expected Deliverables -
i) Output Screenshot (Terminal should have SRN visible) for step 1



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ echo "$(python3 -c 'print("A"*63)')" > prefix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ echo "$(python3 -c 'print("B"*63)')" > suffix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 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: 3a136d35359c6ae88c3b9d2b9747734b

Generating first block: .
Generating second block: S01.............................
Running time: 2.90727 s
[11/12/24]seed@VM:~/.../PES2UG22CS556$ cat out1.bin suffix > file1
[11/12/24]seed@VM:~/.../PES2UG22CS556$ cat out2.bin suffix > file2
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum file1
0a9fe6f424b812b0a0814e330c89fb62  file1
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum file2
0a9fe6f424b812b0a0814e330c89fb62  file2
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 
```

ii) Are the hashes same? Explain your observation.

The MD5 collision was created in out1.bin and out2.bin to intentionally generate the same hash, regardless of their internal differences.

When you append the same suffix to both files, it does not introduce enough change to break the hash collision, so both final files, file1 and file2 still have the same hash.

This result highlights the weakness of MD5 in collision resistance: **two different files can still produce the same hash, even after modifications**.

Problem 3 : Generating Two Executable Files with the Same MD5 Hash

Step 1 : Create the code file [Let us initialise the array with 200 A’s so that it is easy locate 200 A’s in the binary.]

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

Step 2: Compile the code and open the executable in a hex editor

a) gcc task3.c -o task3
b) bless task3

Step 3 : Truncate the prefix and suffix

```
$ head -c 12352 task3 > prefix
$ tail -c +12480 task3 > suffix
```

Step 4 : Generate two files with the prefix and append the suffix to the m to make normal programs. Make them executable.

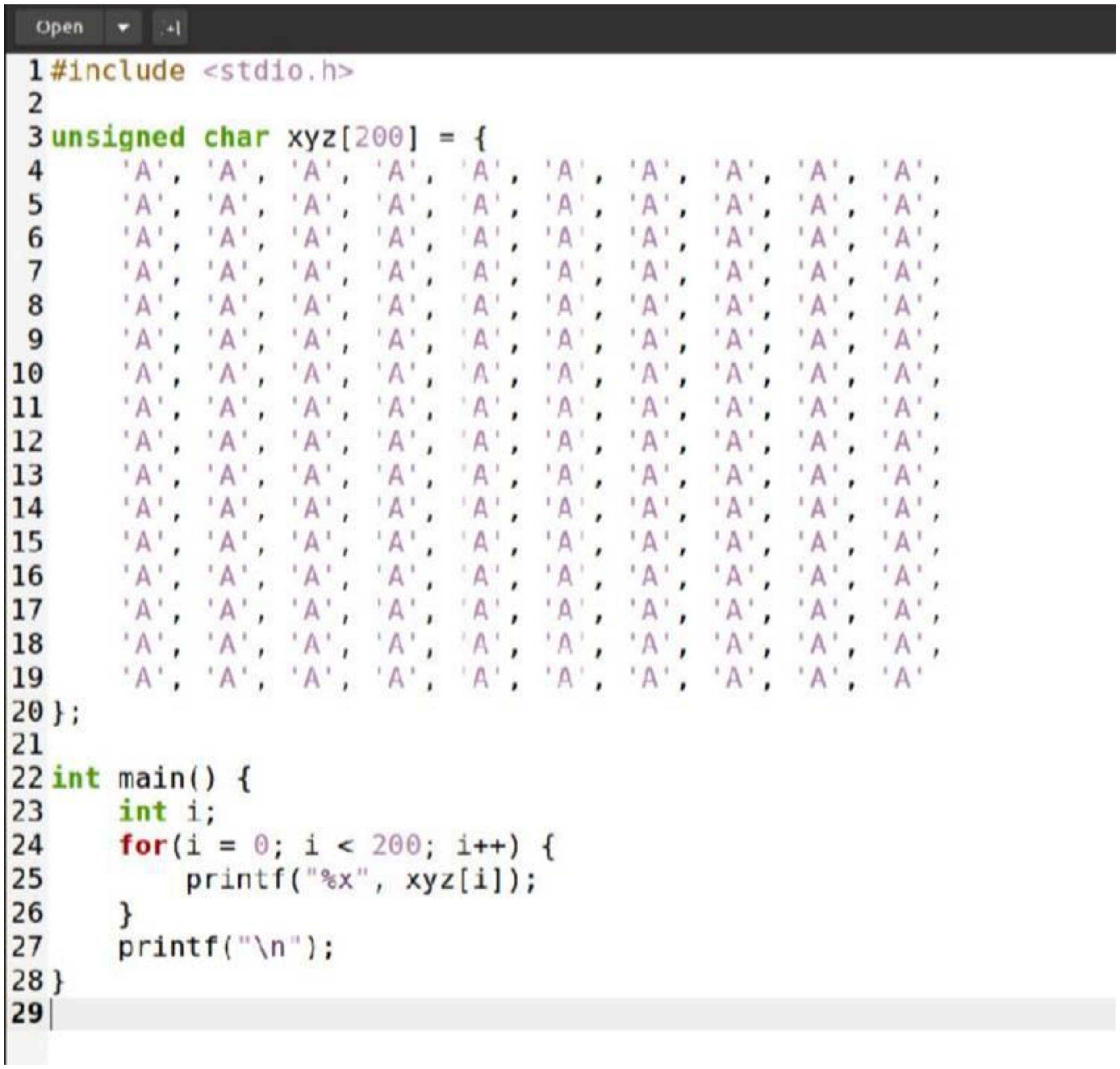
```
$ md5collgen -p prefix -o P Q
$ cat P suffix > arr1.out
$ cat Q suffix > arr2.out
$ sudo chmod +x arr1.out arr2.out
```

Step 5: Verify the success of the task

```
 $ ./arr1.out > f1
$ ./arr2.out > f2
# Compare the md5 sums
$ md5sum arr1.out
$ md5sum arr2.out
# Compare the output of the programs
$ diff f1 f2
```

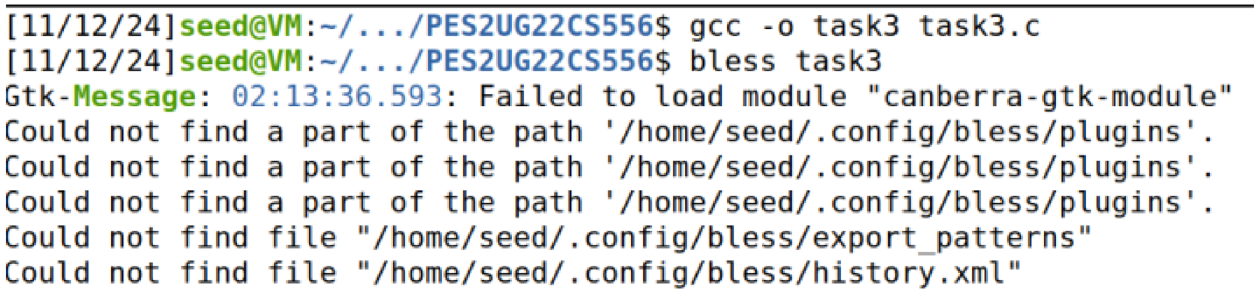
Expected Deliverables -

i) Output Screenshot (Terminal should have SRN visible) for step 1

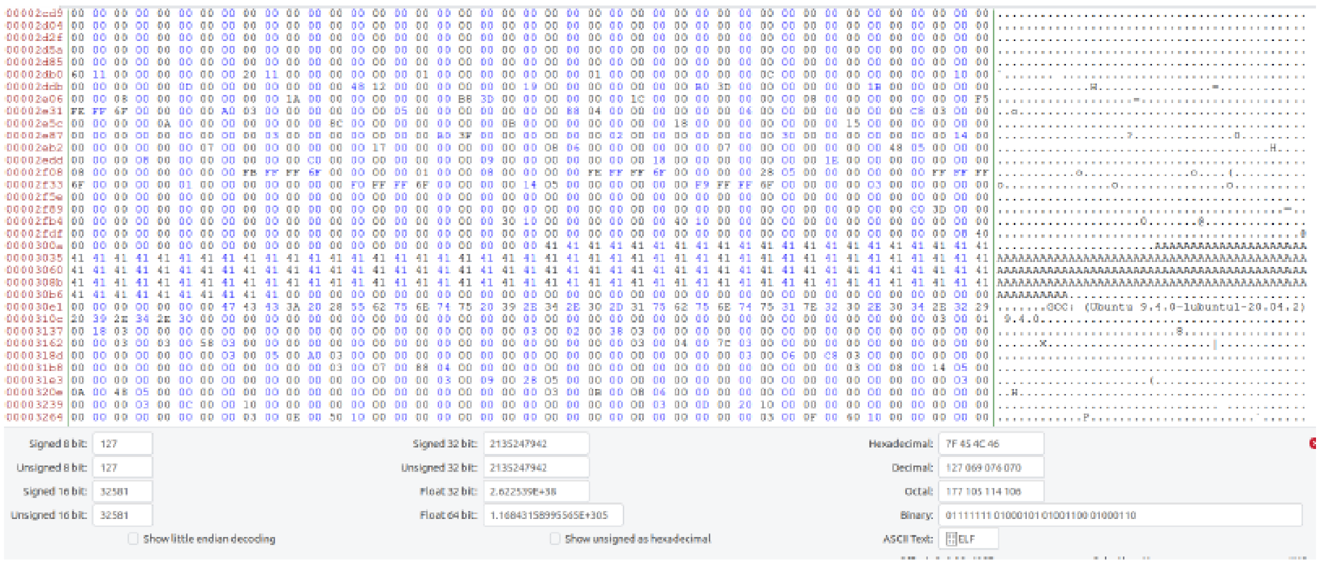


```c
#include <stdio.h>

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");
}
```

ii) Output Screenshot (Terminal should have SRN visible) for step 2.



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ gcc -o task3 task3.c
[11/12/24]seed@VM:~/.../PES2UG22CS556$ bless task3
Gtk-Message: 02:13:36.593: Failed to load module "canberra-gtk-module"
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find file "/home/seed/.config/bless/export_patterns"
Could not find file "/home/seed/.config/bless/history.xml"
```

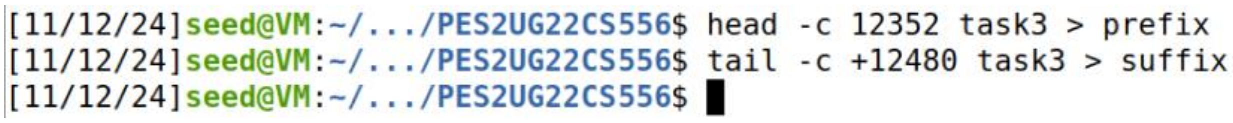


```
00002cd9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002d04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002d2f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002d5a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002d85 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002db0 60 11 00 00 00 00 00 00 20 11 00 00 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 00 00 10 00 `....... ..................................
00002dcb 00 00 00 00 00 0D 00 00 00 00 00 00 00 48 12 00 00 00 00 00 00 19 00 00 00 00 00 00 00 B0 3D 00 00 00 00 00 00 1B 00 00 00 00 00 .............H................=............
00002e06 00 00 08 00 00 00 00 00 00 00 1A 00 00 00 00 00 00 00 B8 3D 00 00 00 00 00 00 1C 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 F5 ...................=.......................
00002e31 FE FF 6F 00 00 00 00 A0 03 00 00 00 00 00 00 05 00 00 00 00 00 00 00 88 04 00 00 00 00 00 00 04 00 00 00 00 00 00 00 C8 03 00 00 ..o........................................
00002e5c 00 00 00 00 0A 00 00 00 00 00 00 00 8C 00 00 00 00 00 00 00 0B 00 00 00 00 00 00 00 18 00 00 00 00 00 00 00 15 00 00 00 00 00 00 ...........................................
00002e87 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 B0 3F 00 00 00 00 00 00 02 00 00 00 00 00 00 00 30 00 00 00 00 00 00 00 14 00 ..................?..............0.........
00002eb2 00 00 00 00 00 00 07 00 00 00 00 00 00 00 17 00 00 00 00 00 00 00 08 06 00 00 00 00 00 00 07 00 00 00 00 00 00 00 48 05 00 00 00 ......................................H....
00002edd 00 00 00 08 00 00 00 00 00 00 00 C0 00 00 00 00 00 00 00 09 00 00 00 00 00 00 00 18 00 00 00 00 00 00 1E 00 00 00 00 00 00 00 00 ...........................................
00002f08 08 00 00 00 00 00 00 00 FB FF FF 6F 00 00 00 00 01 00 00 08 00 00 00 00 FE FF FF 6F 00 00 00 00 28 05 00 00 00 00 00 00 FF FF FF ...........o...............o....(..........
00002f33 6F 00 00 00 00 01 00 00 00 00 00 00 00 F0 FF FF 6F 00 00 00 00 14 05 00 00 00 00 00 00 F9 FF FF 6F 00 00 00 00 03 00 00 00 00 00 o...............o...............o..........
00002f5e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...........................................
00002f89 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 C0 3D 00 00 ........................................=..
00002fb4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 30 10 00 00 00 00 00 00 40 10 00 00 00 00 00 00 00 00 00 00 00 00 00 ....................0.......@..............
00002fdf 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 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 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ......................AAAAAAAAAAAAAAAAAAAAA
00003035 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
00003060 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
0000308b 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
000030b6 41 41 41 41 41 41 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAAAAAAAA.................................
000030e1 00 00 00 00 00 00 00 47 43 43 3A 20 28 55 62 75 6E 74 75 20 39 2E 34 2E 30 2D 31 75 62 75 6E 74 75 31 7E 32 30 2E 30 34 2E 32 29 .......GCC: (Ubuntu 9.4.0-1ubuntu1~20.04.2)
0000310c 20 39 2E 34 2E 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 01  9.4.0.....................................
00003137 00 18 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 02 00 38 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .........................8.................
00003162 00 00 03 00 03 00 58 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 04 00 7C 03 00 00 00 00 00 00 00 00 00 00 00 ......X.......................|............
0000318d 00 00 00 00 00 00 00 03 00 05 00 A0 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 06 00 C8 03 00 00 00 00 00 00 ...........................................
000031b8 00 00 00 00 00 00 00 00 00 00 00 00 03 00 07 00 88 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 08 00 14 05 00 00 ...........................................
000031e3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 09 00 28 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 .....................(.....................
0000320e 0A 00 48 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0E 00 08 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..H........................................
00003239 00 00 00 03 00 0C 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0D 00 20 10 00 00 00 00 00 00 00 00 00 00 ............................... ...........
00003264 00 00 00 00 00 00 00 00 03 00 0E 00 50 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 0F 00 40 10 00 00 00 00 00 00 ............P.......................@......
```

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
Unsigned 16 bit:	32581	Float 64 bit:	1.16843158995565E+305	Binary:	01111111 01000101 01001100 01000110
☐ Show little endian decoding		☐ Show unsigned as hexadecimal		ASCII Text:	ELF

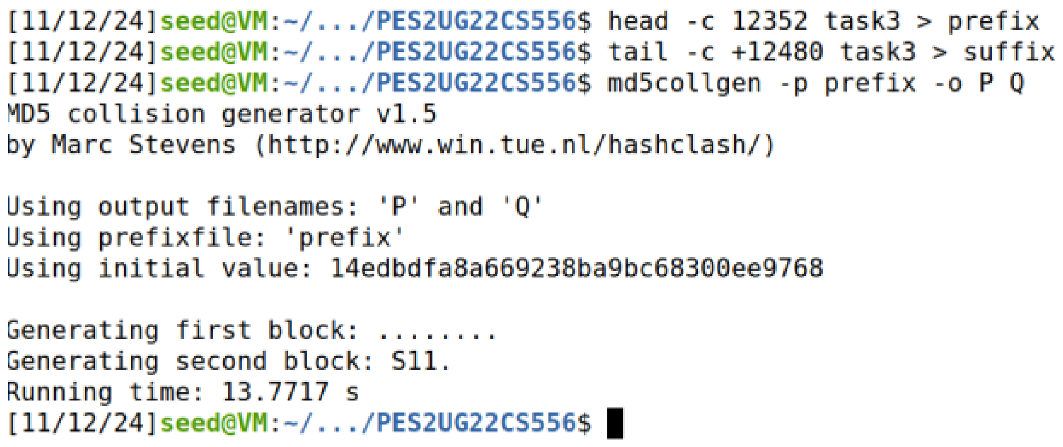
Important: Highlight the starting end ending hexadecimal offsets

iii) Output Screenshot (Terminal should have SRN visible) for step 3

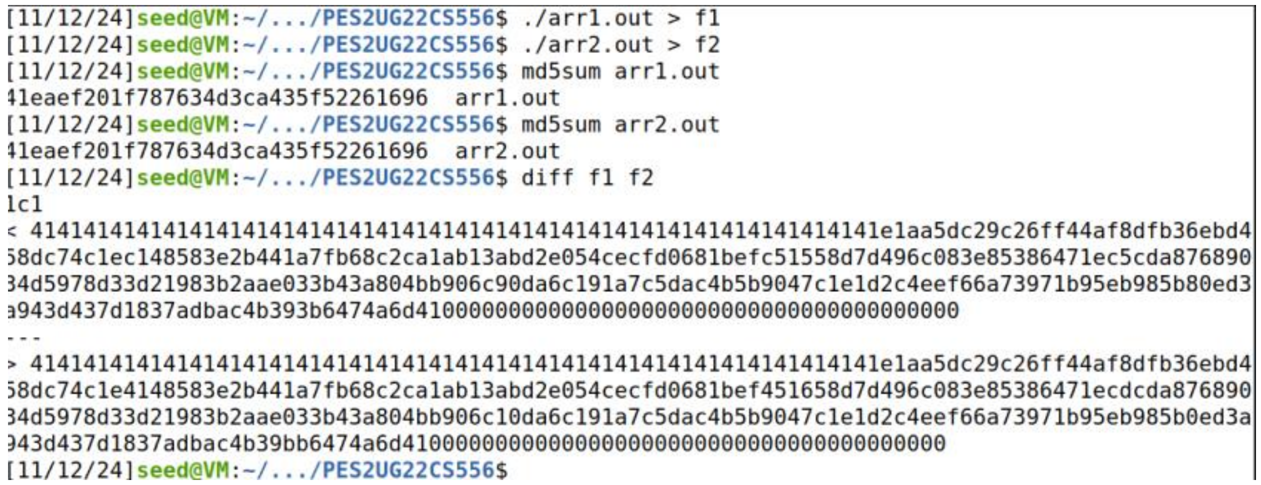


```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ head -c 12352 task3 > prefix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ tail -c +12480 task3 > suffix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 
```

iv) Output Screenshot (Terminal should have SRN visible) for step 4



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ head -c 12352 task3 > prefix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ tail -c +12480 task3 > suffix
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5collgen -p prefix -o P Q
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'P' and 'Q'
Using prefixfile: 'prefix'
Using initial value: 14edbdfa8a669238ba9bc68300ee9768

Generating first block: ........
Generating second block: S11.
Running time: 13.7717 s
[11/12/24]seed@VM:~/.../PES2UG22CS556$ 
```



```
[11/12/24]seed@VM:~/.../PES2UG22CS556$ ./arr1.out > f1
[11/12/24]seed@VM:~/.../PES2UG22CS556$ ./arr2.out > f2
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum arr1.out
41eaef201f787634d3ca435f52261696  arr1.out
[11/12/24]seed@VM:~/.../PES2UG22CS556$ md5sum arr2.out
41eaef201f787634d3ca435f52261696  arr2.out
[11/12/24]seed@VM:~/.../PES2UG22CS556$ diff f1 f2
1c1
< 4141414141414141414141414141414141414141414141414141414141414141e1aa5dc29c26ff44af8dfb36ebd4
58dc74c1ec148583e2b441a7fb68c2ca1ab13abd2e054cecfd0681befc51558d7d496c083e85386471ec5cda876890
34d5978d33d21983b2aae033b43a804bb906c90da6c191a7c5dac4b5b9047c1e1d2c4eef66a73971b95eb985b80ed3
a943d437d1837adbac4b393b6474a6d41000000000000000000000000000000000000000
---
> 4141414141414141414141414141414141414141414141414141414141414141e1aa5dc29c26ff44af8dfb36ebd4
58dc74c1e4148583e2b441a7fb68c2ca1ab13abd2e054cecfd0681bef451658d7d496c083e85386471ecdcda876890
34d5978d33d21983b2aae033b43a804bb906c10da6c191a7c5dac4b5b9047c1e1d2c4eef66a73971b95eb985b0ed3a
943d437d1837adbac4b39bb6474a6d41000000000000000000000000000000000000000
[11/12/24]seed@VM:~/.../PES2UG22CS556$
```

v) Explain observations

- I observed that file1 and file2 produced the same MD5 hash even though they had different content.
- This happened because the MD5 algorithm is vulnerable to collision attacks, which allow two different inputs to generate the same hash.
- The md5collgen tool exploits this by manipulating a specific part (128-byte region) of the files, causing them to differ while maintaining the same hash.
- This shows why MD5 is no longer secure for verifying file integrity or for cryptographic purposes.