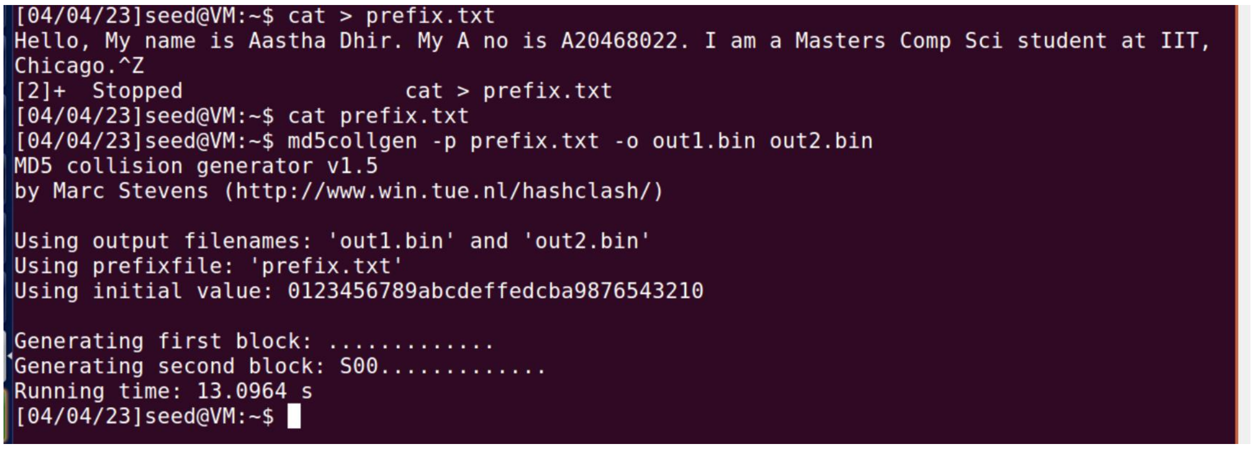
Assignment 3- CS458

Task 1 Generating two different files with the same MD5 Hash

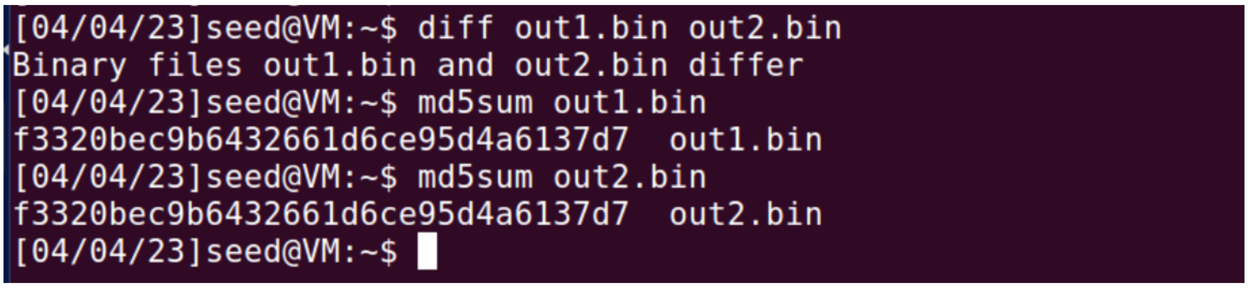
In this task, we have generated two different files with the same MD5 hash values. The beginning parts of these two files are the same, i.e., they share the same prefix. We have achieved this using the md5collgen program. The following command generated two output files, out1.bin and out2.bin, for a given prefix file prefix.txt:

$ md5collgen -p prefix.txt -o out1.bin out2.bin



```
[04/04/23]seed@VM:~$ cat > prefix.txt
Hello, My name is Aastha Dhir. My A no is A20468022. I am a Masters Comp Sci student at IIT,
Chicago.^Z
[2]+  Stopped                 cat > prefix.txt
[04/04/23]seed@VM:~$ cat prefix.txt
[04/04/23]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: 0123456789abcdeffedcba9876543210

Generating first block: ............
Generating second block: S00............
Running time: 13.0964 s
[04/04/23]seed@VM:~$
```

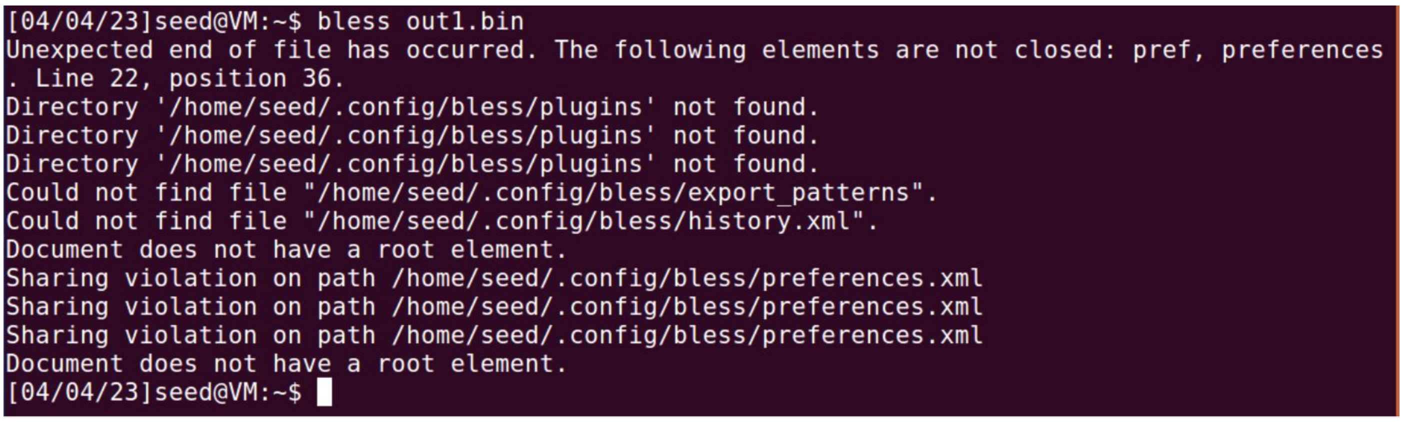
We looked at the differences between the output files using the diff command. We also used the md5sum command to check the MD5 hash of each output file.



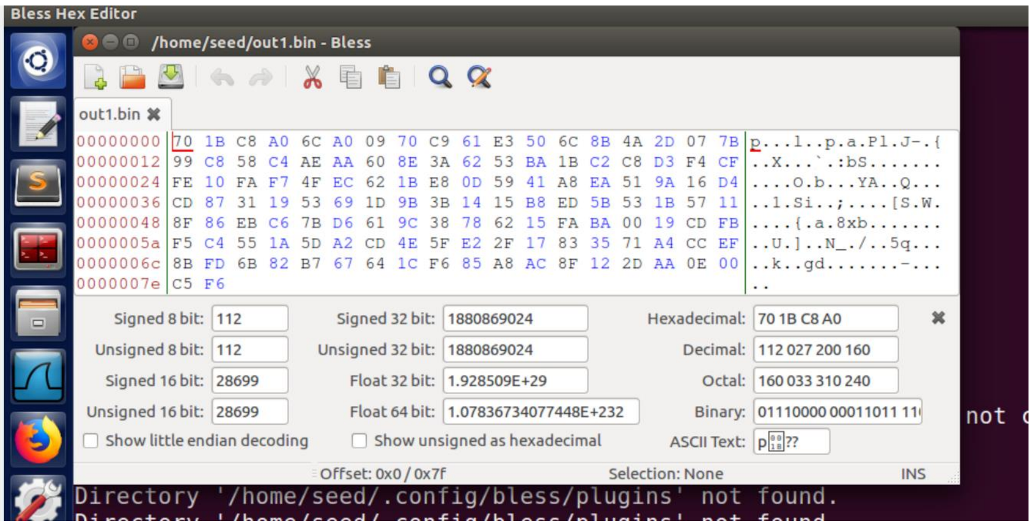
```
[04/04/23]seed@VM:~$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[04/04/23]seed@VM:~$ md5sum out1.bin
f3320bec9b6432661d6ce95d4a6137d7  out1.bin
[04/04/23]seed@VM:~$ md5sum out2.bin
f3320bec9b6432661d6ce95d4a6137d7  out2.bin
[04/04/23]seed@VM:~$
```

Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?

If the length of our prefix file is not a multiple of 64, zeros will be padded to the file. This is because MD5 processes the file in blocks of size 64 bytes. From the screenshot given below, we can see that zeros were padded to the file because the file size is not a multiple of 64.



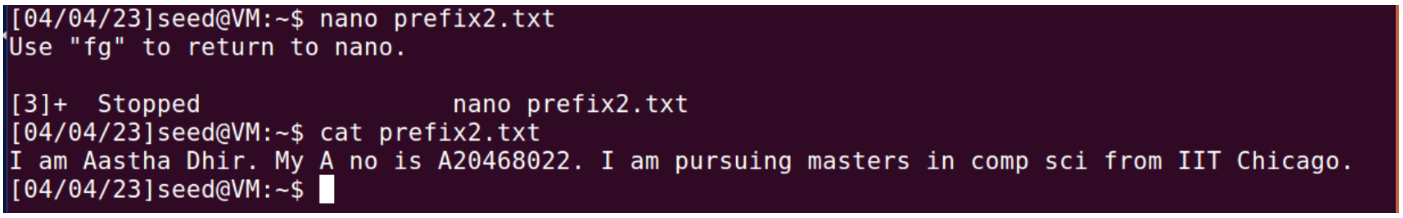
```
[04/04/23]seed@VM:~$ bless out1.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences
. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[04/04/23]seed@VM:~$
```



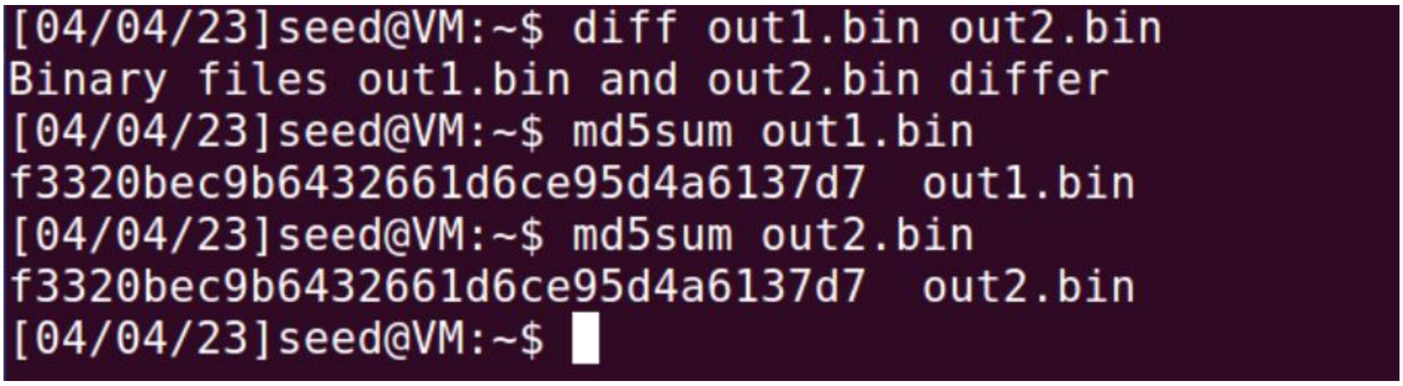
```
/home/seed/out1.bin - Bless
out1.bin
00000000 70 1B C8 A0 6C A0 09 70 C9 61 E3 50 6C 8B 4A 2D 07 7B p...l..p.a.Pl.J-.{
00000012 99 C8 58 C4 AE AA 60 8E 3A 62 53 BA 1B C2 C8 D3 F4 CF ..X...`.:bS.......
00000024 FE 10 FA F7 4F EC 62 1B E8 0D 59 41 A8 EA 51 9A 16 D4 ....O.b...YA..Q...
00000036 CD 87 31 19 53 69 1D 9B 3B 14 15 B8 ED 5B 53 1B 57 11 ..1.Si..;....[S.W.
00000048 8F 86 EB C6 7B D6 61 9C 38 78 62 15 FA BA 00 19 CD FB ....{.a.8xb.......
0000005a F5 C4 55 1A 5D A2 CD 4E 5F E2 2F 17 83 35 71 A4 CC EF ..U.]..N_./..5q...
0000006c 8B FD 6B 82 B7 67 64 1C F6 85 A8 AC 8F 12 2D AA 0E 00 ..k..gd.......-...
0000007e C5 F6                                                ..

Signed 8 bit: 112        Signed 32 bit: 1880869024            Hexadecimal: 70 1B C8 A0
Unsigned 8 bit: 112      Unsigned 32 bit: 1880869024          Decimal: 112 027 200 160
Signed 16 bit: 28699     Float 32 bit: 1.928509E+29           Octal: 160 033 310 240
Unsigned 16 bit: 28699   Float 64 bit: 1.07836734077448E+232  Binary: 01110000 00011011 11
Show little endian decoding   Show unsigned as hexadecimal    ASCII Text: p??
Offset: 0x0 / 0x7f            Selection: None                 INS
Directory '/home/seed/.config/bless/plugins' not found.
```

Question 2. Create a prefix file with exactly 64 bytes, run the collision tool again, and see what happens.

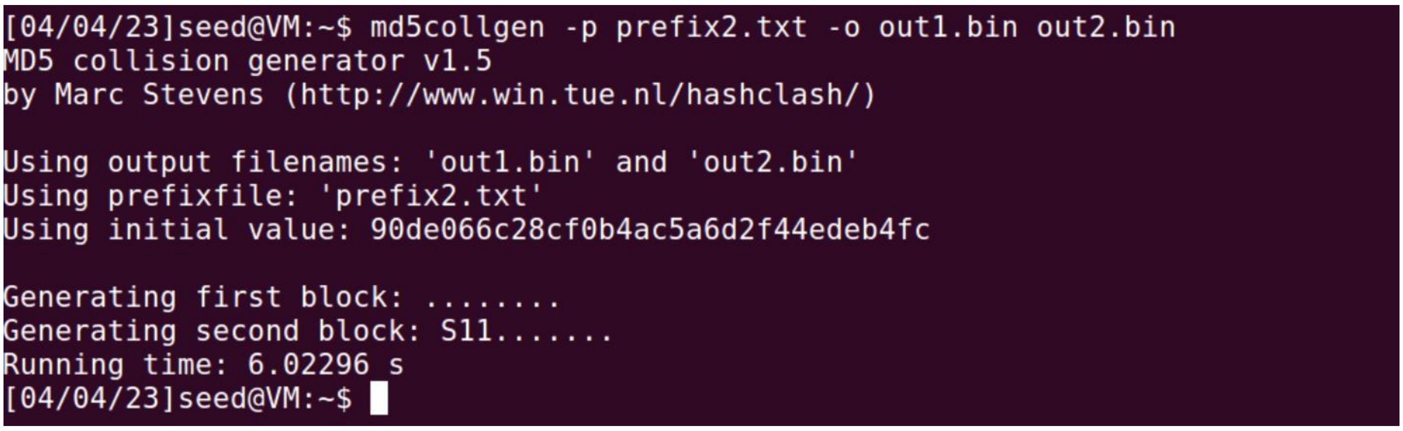
From the screenshots given below, we see that when you create a file with a 64-byte prefix, there are no extra zeros padded to the file. The file is exactly 64 bytes in size in the bless editor.



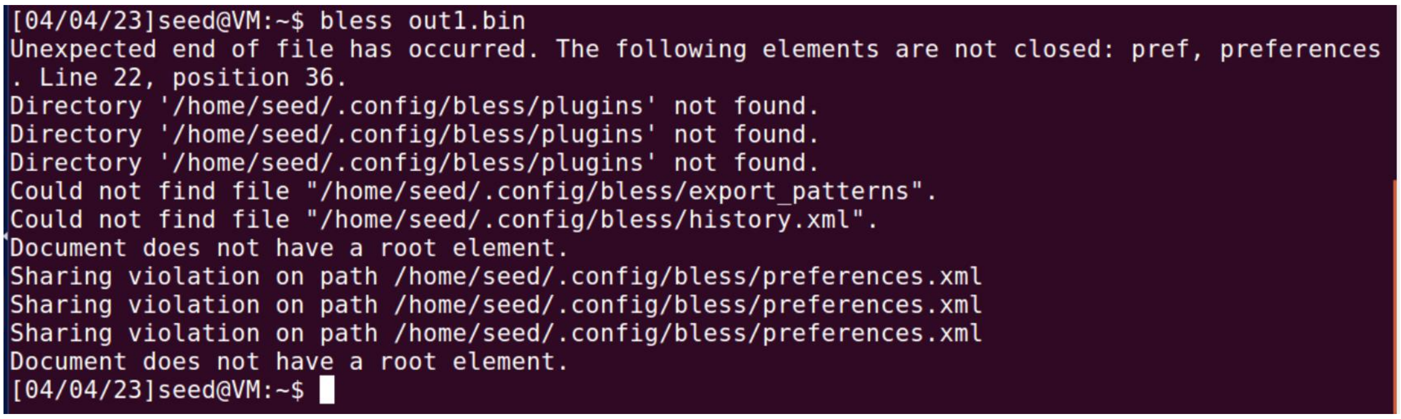
```
[04/04/23]seed@VM:~$ nano prefix2.txt
Use "fg" to return to nano.

[3]+  Stopped                 nano prefix2.txt
[04/04/23]seed@VM:~$ cat prefix2.txt
I am Aastha Dhir. My A no is A20468022. I am pursuing masters in comp sci from IIT Chicago.
[04/04/23]seed@VM:~$
```



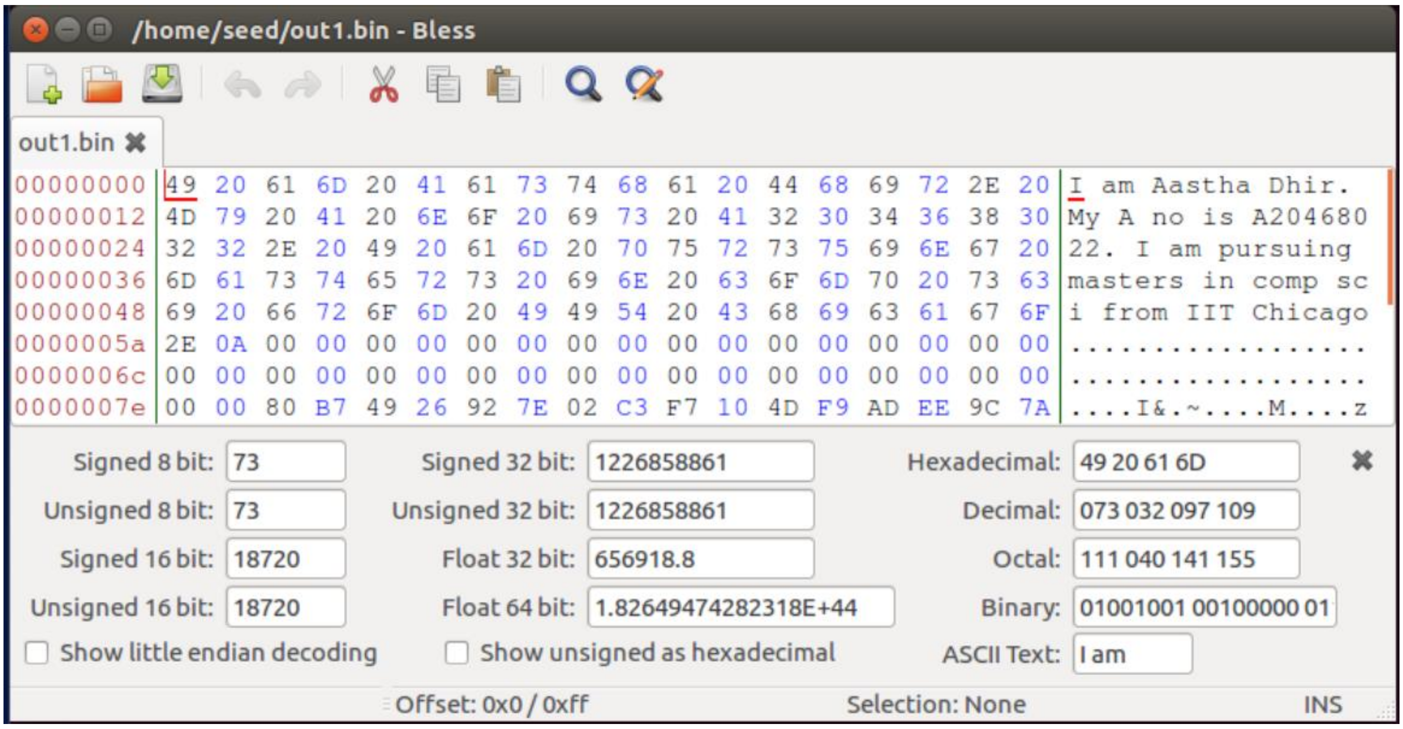
```
[04/04/23]seed@VM:~$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[04/04/23]seed@VM:~$ md5sum out1.bin
f3320bec9b6432661d6ce95d4a6137d7  out1.bin
[04/04/23]seed@VM:~$ md5sum out2.bin
f3320bec9b6432661d6ce95d4a6137d7  out2.bin
[04/04/23]seed@VM:~$
```



```
[04/04/23]seed@VM:~$ md5collgen -p prefix2.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: 'prefix2.txt'
Using initial value: 90de066c28cf0b4ac5a6d2f44edeb4fc

Generating first block: ........
Generating second block: S11.......
Running time: 6.02296 s
[04/04/23]seed@VM:~$
```



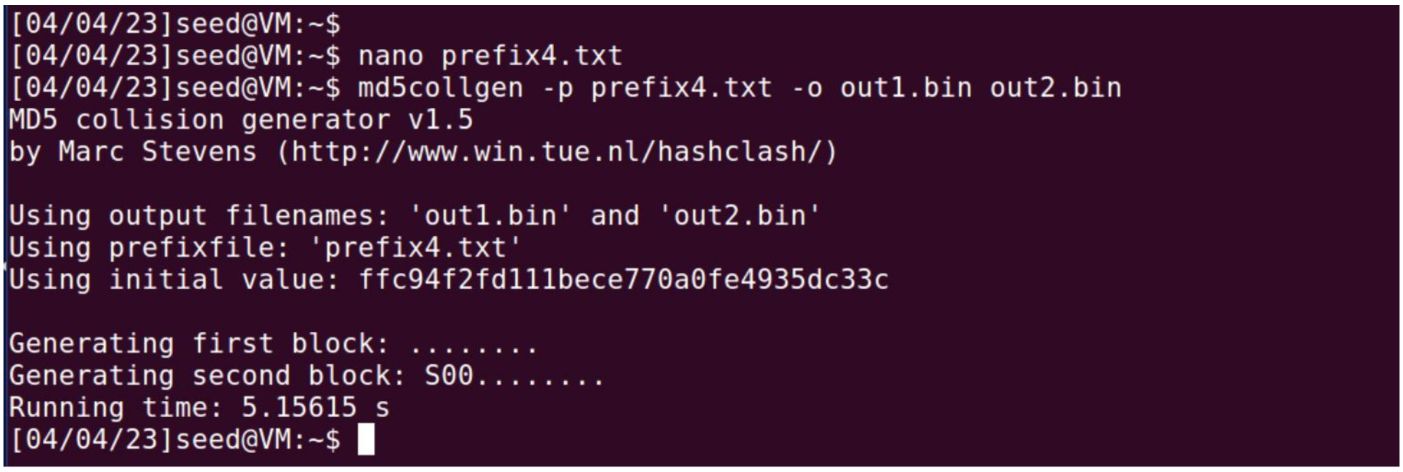
```
[04/04/23]seed@VM:~$ bless out1.bin
Unexpected end of file has occurred. The following elements are not closed: pref, preferences
. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[04/04/23]seed@VM:~$
```



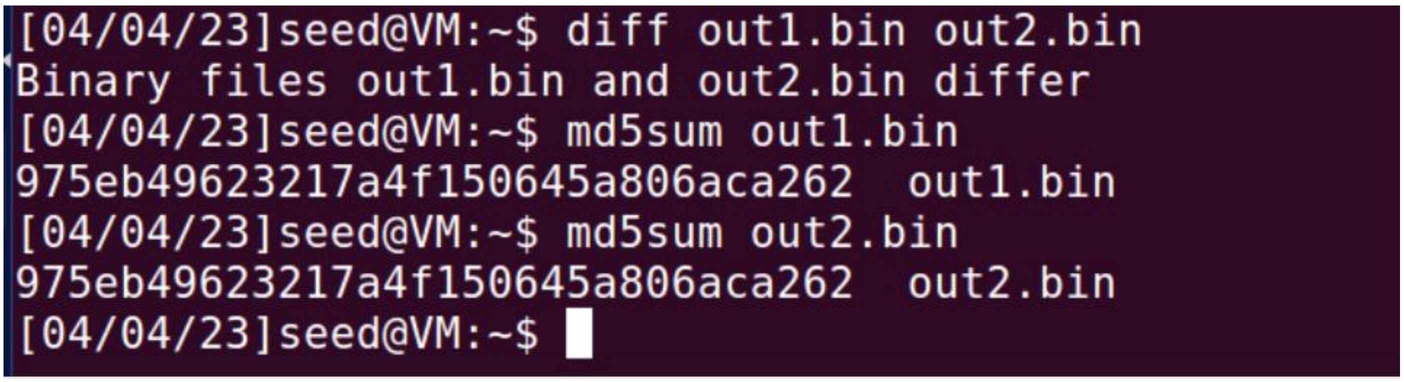
```
/home/seed/out1.bin - Bless
out1.bin
00000000 49 20 61 6D 20 41 61 73 74 68 61 20 44 68 69 72 2E 20 I am Aastha Dhir.
00000012 4D 79 20 41 20 6E 6F 20 69 73 20 41 32 30 34 36 38 30 My A no is A204680
00000024 32 32 2E 20 49 20 61 6D 20 70 75 72 73 75 69 6E 67 20 22. I am pursuing
00000036 6D 61 73 74 65 72 73 20 69 6E 20 63 6F 6D 70 20 73 63 masters in comp sc
00000048 69 20 66 72 6F 6D 20 49 49 54 20 43 68 69 63 61 67 6F i from IIT Chicago
0000005a 2E 0A 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 00 00 00 00 00 00 00 00 00 00 ..................
0000007e 00 00 80 B7 49 26 92 7E 02 C3 F7 10 4D F9 AD EE 9C 7A ....I&.~....M....z

Signed 8 bit: 73          Signed 32 bit: 1226858861      Hexadecimal: 49 20 61 6D
Unsigned 8 bit: 73        Unsigned 32 bit: 1226858861    Decimal: 073 032 097 109
Signed 16 bit: 18720      Float 32 bit: 656918.8         Octal: 111 040 141 155
Unsigned 16 bit: 18720    Float 64 bit: 1.82649474282318E+44   Binary: 01001001 00100000 01
Show little endian decoding   Show unsigned as hexadecimal   ASCII Text: I am
Offset: 0x0 / 0xff        Selection: None        INS
```

Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

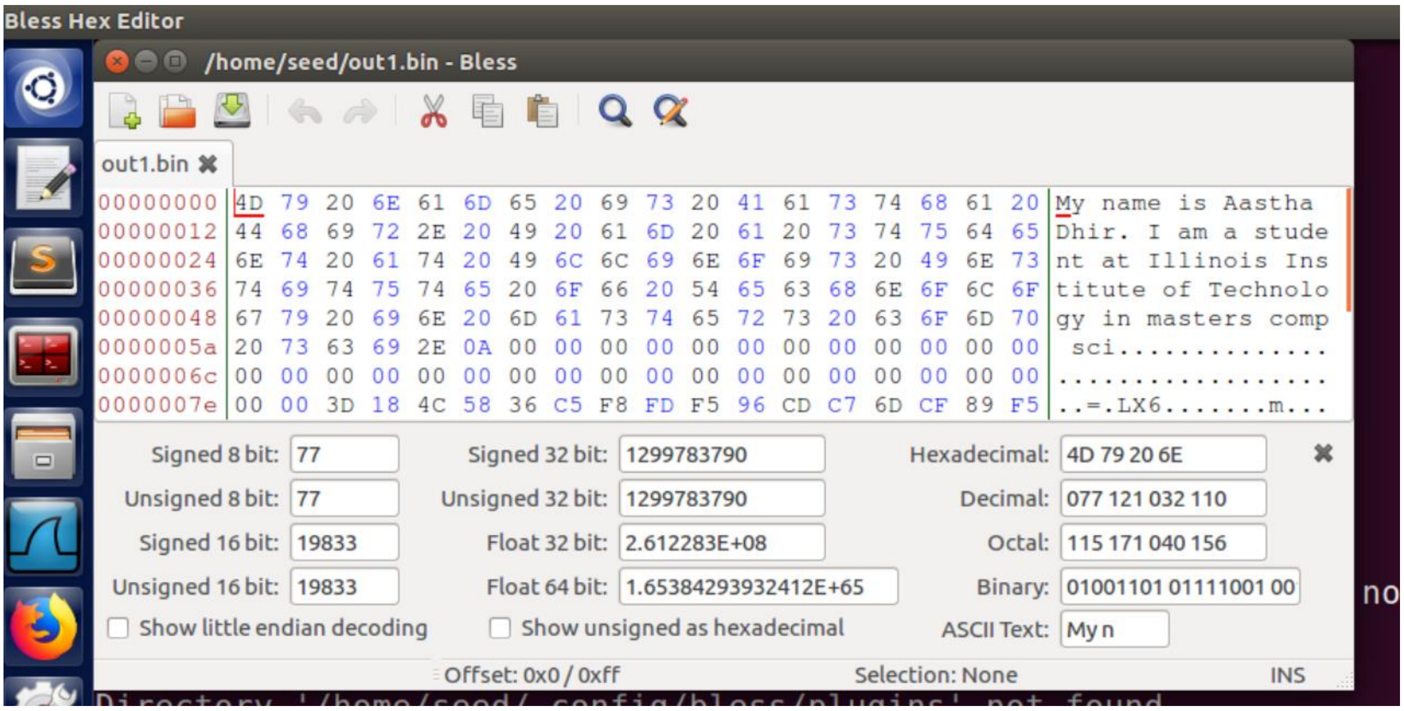
The data that is generated is not completely different for the two output files. We observe that only a few bytes differ in both files.



```
[04/04/23]seed@VM:~$
[04/04/23]seed@VM:~$ nano prefix4.txt
[04/04/23]seed@VM:~$ md5collgen -p prefix4.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: 'prefix4.txt'
Using initial value: ffc94f2fd111bece770a0fe4935dc33c

Generating first block: ........
Generating second block: S00........
Running time: 5.15615 s
[04/04/23]seed@VM:~$
```



```
[04/04/23]seed@VM:~$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[04/04/23]seed@VM:~$ md5sum out1.bin
975eb49623217a4f150645a806aca262  out1.bin
[04/04/23]seed@VM:~$ md5sum out2.bin
975eb49623217a4f150645a806aca262  out2.bin
[04/04/23]seed@VM:~$ 
```



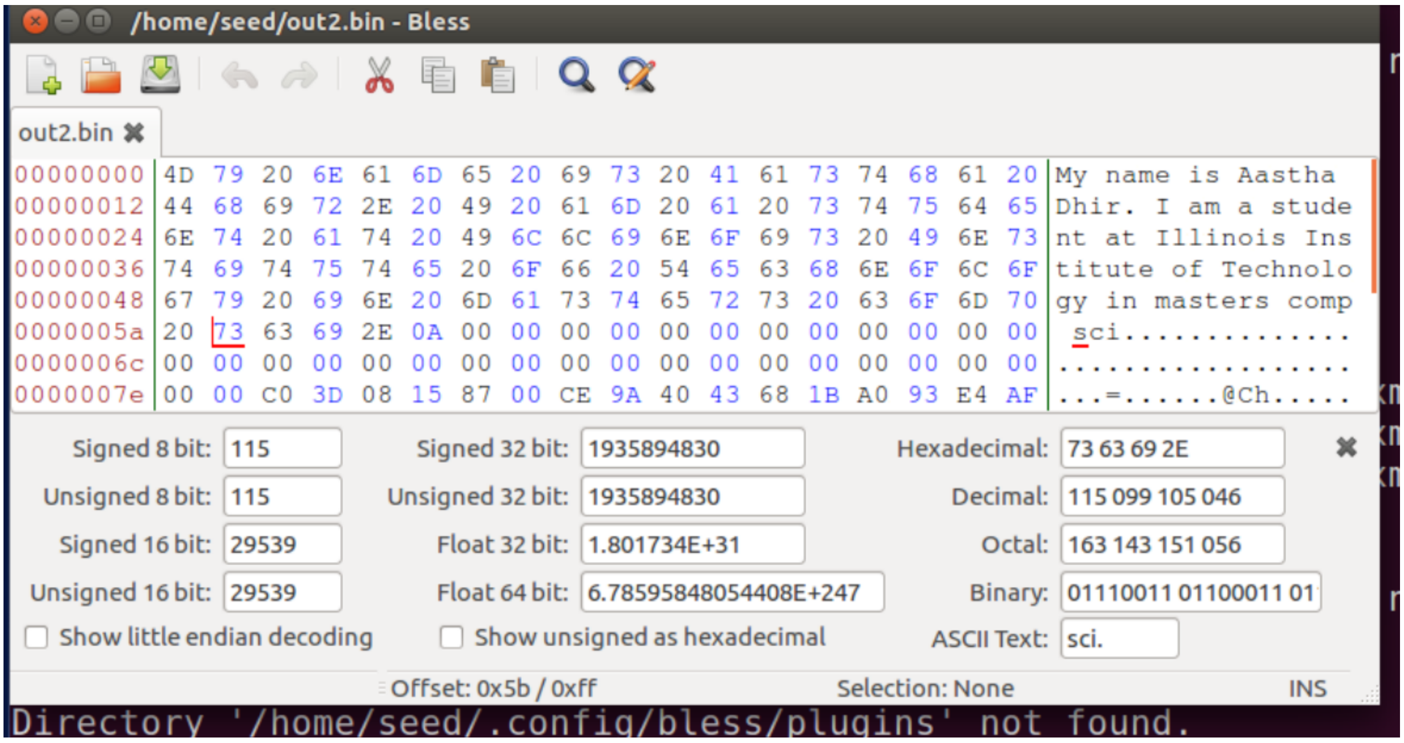
Bless Hex Editor

/home/seed/out1.bin - Bless

out1.bin

```
00000000 4D 79 20 6E 61 6D 65 20 69 73 20 41 61 73 74 68 61 20 My name is Aastha 
00000012 44 68 69 72 2E 20 49 20 61 6D 20 61 20 73 74 75 64 65 Dhir. I am a stude
00000024 6E 74 20 61 74 20 49 6C 6C 69 6E 6F 69 73 20 49 6E 73 nt at Illinois Ins
00000036 74 69 74 75 74 65 20 6F 66 20 54 65 63 68 6E 6F 6C 6F titute of Technolo
00000048 67 79 20 69 6E 20 6D 61 73 74 65 72 73 20 63 6F 6D 70 gy in masters comp
0000005a 20 73 63 69 2E 0A 00 00 00 00 00 00 00 00 00 00 00 00  sci..............
0000006c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000007e 00 00 3D 18 4C 58 36 C5 F8 FD F5 96 CD C7 6D CF 89 F5 ..=.LX6.......m...
```

```
Signed 8 bit: 77               Signed 32 bit: 1299783790               Hexadecimal: 4D 79 20 6E
Unsigned 8 bit: 77             Unsigned 32 bit: 1299783790             Decimal: 077 121 032 110
Signed 16 bit: 19833           Float 32 bit: 2.612283E+08              Octal: 115 171 040 156
Unsigned 16 bit: 19833         Float 64 bit: 1.65384293932412E+65      Binary: 01001101 01111001 00
Show little endian decoding    Show unsigned as hexadecimal            ASCII Text: My n
Offset: 0x0 / 0xff             Selection: None                         INS
```



/home/seed/out2.bin - Bless

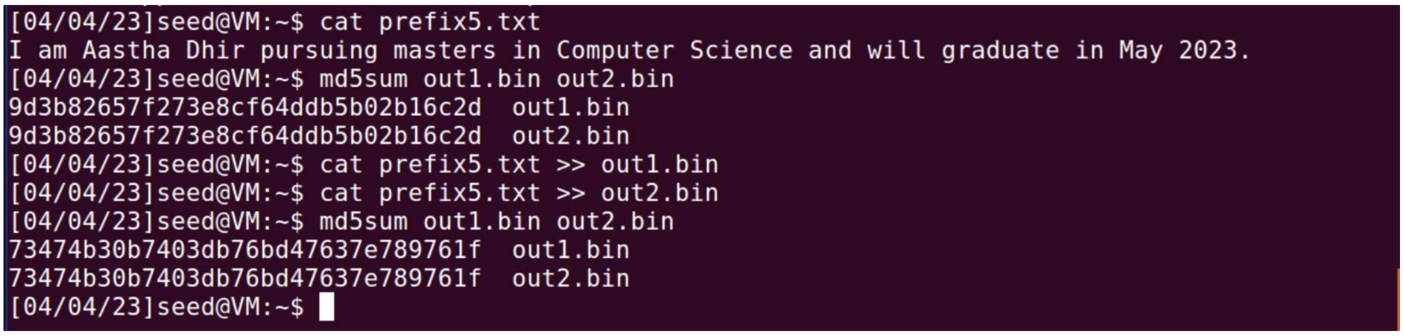
out2.bin

```
00000000 4D 79 20 6E 61 6D 65 20 69 73 20 41 61 73 74 68 61 20 My name is Aastha 
00000012 44 68 69 72 2E 20 49 20 61 6D 20 61 20 73 74 75 64 65 Dhir. I am a stude
00000024 6E 74 20 61 74 20 49 6C 6C 69 6E 6F 69 73 20 49 6E 73 nt at Illinois Ins
00000036 74 69 74 75 74 65 20 6F 66 20 54 65 63 68 6E 6F 6C 6F titute of Technolo
00000048 67 79 20 69 6E 20 6D 61 73 74 65 72 73 20 63 6F 6D 70 gy in masters comp
0000005a 20 73 63 69 2E 0A 00 00 00 00 00 00 00 00 00 00 00 00  sci..............
0000006c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000007e 00 00 C0 3D 08 15 87 00 CE 9A 40 43 68 1B A0 93 E4 AF ...=......@Ch.....
```

```
Signed 8 bit: 115              Signed 32 bit: 1935894830               Hexadecimal: 73 63 69 2E
Unsigned 8 bit: 115            Unsigned 32 bit: 1935894830             Decimal: 115 099 105 046
Signed 16 bit: 29539           Float 32 bit: 1.801734E+31              Octal: 163 143 151 056
Unsigned 16 bit: 29539         Float 64 bit: 6.78595848054408E+247     Binary: 01110011 01100011 01
Show little endian decoding    Show unsigned as hexadecimal            ASCII Text: sci.
Offset: 0x5b / 0xff            Selection: None                         INS
Directory '/home/seed/.config/bless/plugins' not found.
```

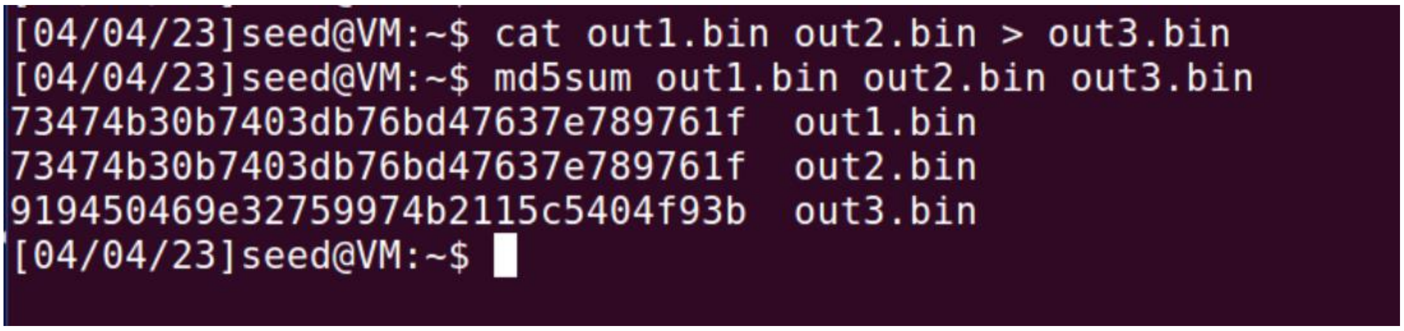
Task 2: Understanding MD5's Property

We are going to create a file prefix.txt and check to see if the MD5 hashes of the generated files are the same. After that, we will randomly add a string to the end of both files out1.bin and out2.bin, and check their MD5 hashes again.



```
[04/04/23]seed@VM:~$ cat prefix5.txt
I am Aastha Dhir pursuing masters in Computer Science and will graduate in May 2023.
[04/04/23]seed@VM:~$ md5sum out1.bin out2.bin
9d3b82657f273e8cf64ddb5b02b16c2d  out1.bin
9d3b82657f273e8cf64ddb5b02b16c2d  out2.bin
[04/04/23]seed@VM:~$ cat prefix5.txt >> out1.bin
[04/04/23]seed@VM:~$ cat prefix5.txt >> out2.bin
[04/04/23]seed@VM:~$ md5sum out1.bin out2.bin
73474b30b7403db76bd47637e789761f  out1.bin
73474b30b7403db76bd47637e789761f  out2.bin
[04/04/23]seed@VM:~$
```

The new MD5 hashes are different from the old ones, but they're the same because MD5 can be tricked into thinking that a string has been lengthened. Since the MD5 hashes for both files are the same, we can assume that the data within the files was the same after the MD5 algorithm was run.



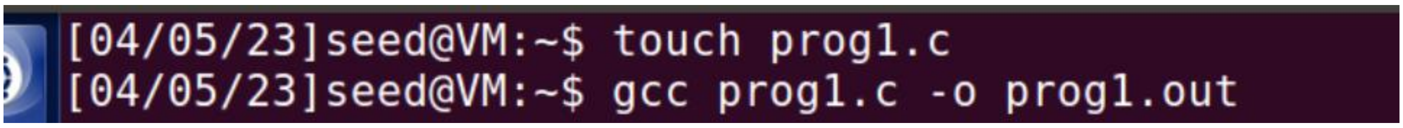
```
[04/04/23]seed@VM:~$ cat out1.bin out2.bin > out3.bin
[04/04/23]seed@VM:~$ md5sum out1.bin out2.bin out3.bin
73474b30b7403db76bd47637e789761f  out1.bin
73474b30b7403db76bd47637e789761f  out2.bin
919450469e32759974b2115c5404f93b  out3.bin
[04/04/23]seed@VM:~$
```

2.3 Task 3: Generating Two Executable Files with the Same MD5 Hash

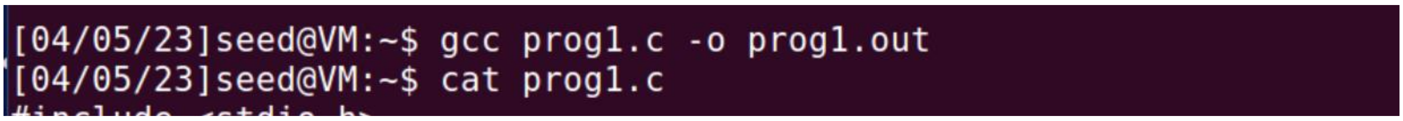
Writing and compiling C program

```
#include <stdio.h>

unsigned char arr[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',

'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',

'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'

/* the actual contents of the array are upto you */

};
```

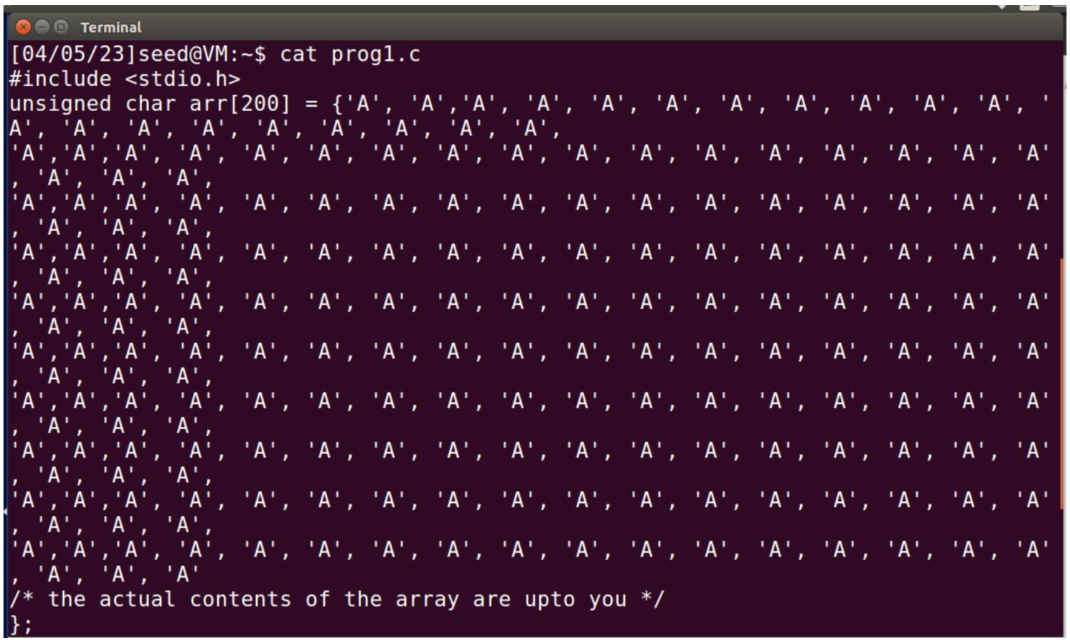
```
int main()

{

int i;

arr[195]='K';

arr[196]='K';

arr[197]='K';

for (i=0; i<200; i++){

printf("%x", arr[i]);

}

printf("\n");

}
```



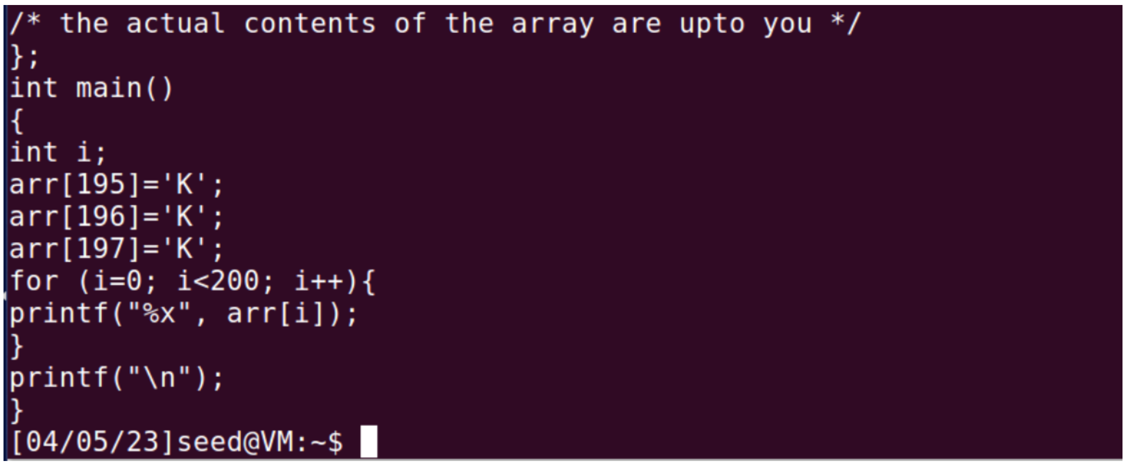
```
[04/05/23]seed@VM:~$ touch prog1.c
[04/05/23]seed@VM:~$ gcc prog1.c -o prog1.out
```



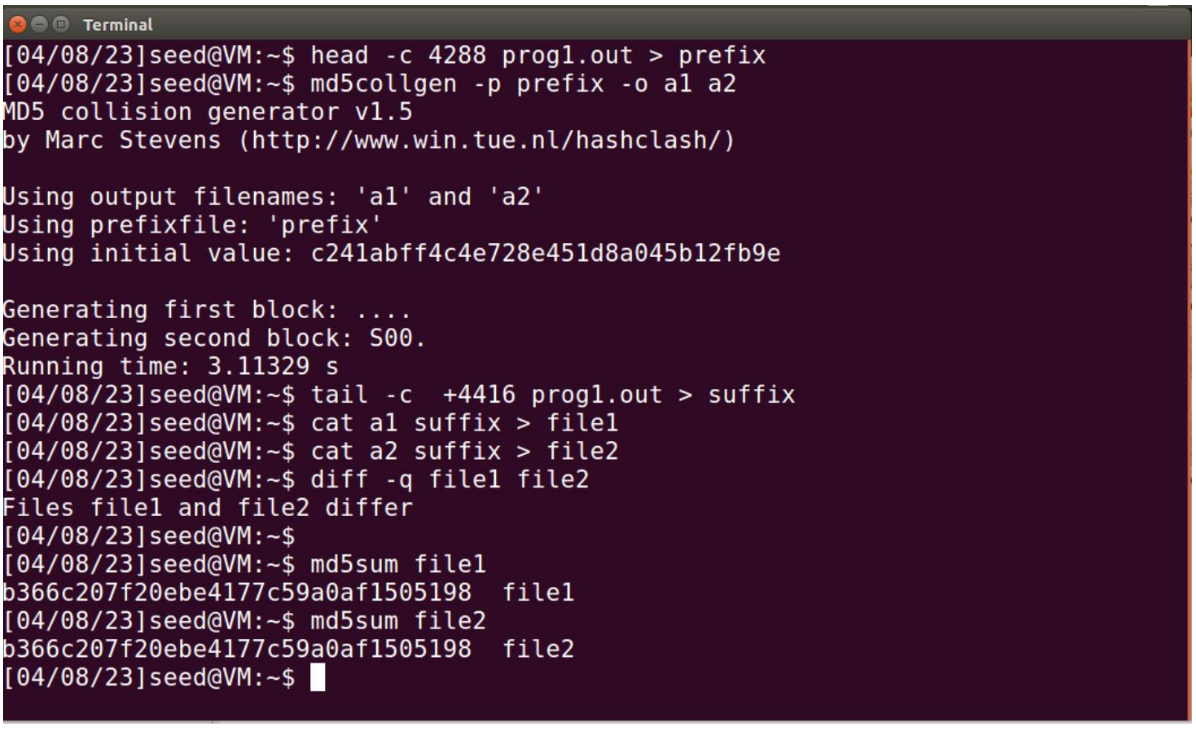
```
[04/05/23]seed@VM:~$ gcc prog1.c -o prog1.out
[04/05/23]seed@VM:~$ cat prog1.c
```



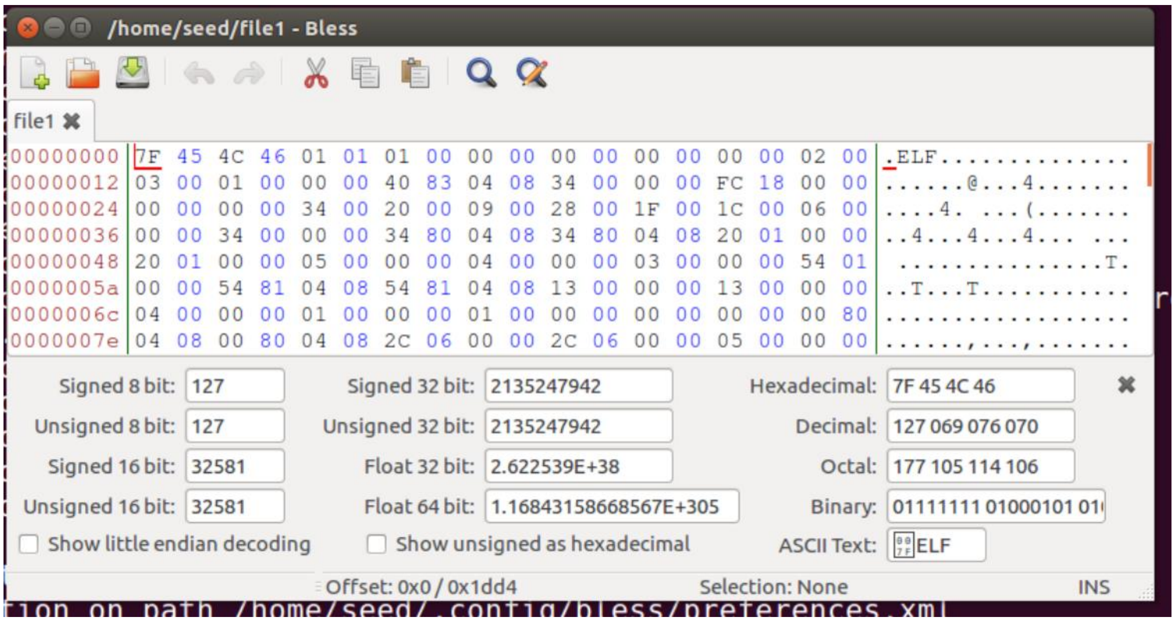
```
[04/05/23]seed@VM:~$ cat prog1.c
#include <stdio.h>
unsigned char arr[200] = {'A', 'A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', '
A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'
, 'A', 'A', 'A',
'A','A','A', 'A', 'A', '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',
'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'
, 'A', 'A', 'A',
'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'
, 'A', 'A', 'A'
/* the actual contents of the array are upto you */
};
```



```
/* the actual contents of the array are upto you */
};
int main()
{
int i;
arr[195]='K';
arr[196]='K';
arr[197]='K';
for (i=0; i<200; i++){
printf("%x", arr[i]);
}
printf("\n");
}
[04/05/23]seed@VM:~$
```

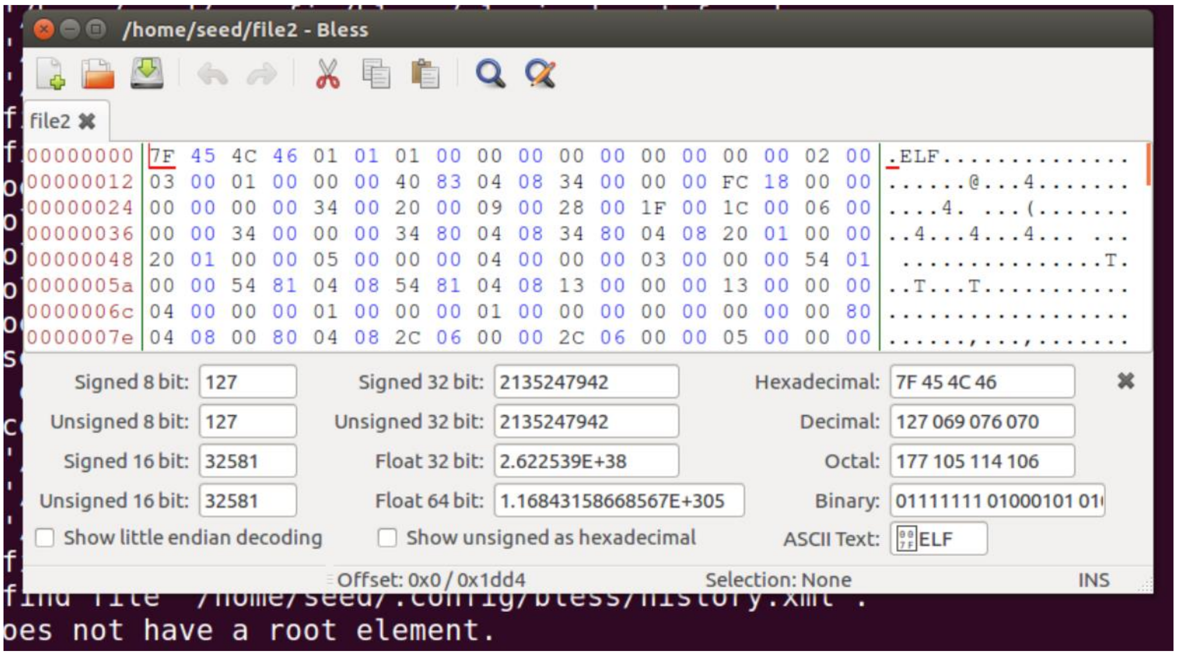


```
[04/08/23]seed@VM:~$ head -c 4288 prog1.out > prefix
[04/08/23]seed@VM:~$ md5collgen -p prefix -o a1 a2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'a1' and 'a2'
Using prefixfile: 'prefix'
Using initial value: c241abff4c4e728e451d8a045b12fb9e

Generating first block: ....
Generating second block: S00.
Running time: 3.11329 s
[04/08/23]seed@VM:~$ tail -c  +4416 prog1.out > suffix
[04/08/23]seed@VM:~$ cat a1 suffix > file1
[04/08/23]seed@VM:~$ cat a2 suffix > file2
[04/08/23]seed@VM:~$ diff -q file1 file2
Files file1 and file2 differ
[04/08/23]seed@VM:~$
[04/08/23]seed@VM:~$ md5sum file1
b366c207f20ebe4177c59a0af1505198  file1
[04/08/23]seed@VM:~$ md5sum file2
b366c207f20ebe4177c59a0af1505198  file2
[04/08/23]seed@VM:~$
```



```
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 02 00 .ELF..............
00000012 03 00 01 00 00 00 40 83 04 08 34 00 00 00 FC 18 00 00 ......@...4.......
00000024 00 00 00 00 34 00 20 00 09 00 28 00 1F 00 1C 00 06 00 ....4. ...(.......
00000036 00 00 34 00 00 00 34 80 04 08 34 80 04 08 20 01 00 00 ..4...4...4... ...
00000048 20 01 00 00 05 00 00 00 04 00 00 00 03 00 00 00 54 01  ..............T.
0000005a 00 00 54 81 04 08 54 81 04 08 13 00 00 00 13 00 00 00 ..T...T...........
0000006c 04 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 80 ..................
0000007e 04 08 00 80 04 08 2C 06 00 00 2C 06 00 00 05 00 00 00 ......,...,.......
```

```
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: 1.16843158668567E+305   Binary: 01111111 01000101 01
Show little endian decoding  Show unsigned as hexadecimal          ASCII Text: ELF
Offset: 0x0 / 0x1dd4         Selection: None                       INS
tion on path /home/seed/.config/bless/preferences.xml
```



```
/home/seed/file2 - Bless
file2
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 02 00 .ELF..............
00000012 03 00 01 00 00 00 40 83 04 08 34 00 00 00 FC 18 00 00 ......@...4.......
00000024 00 00 00 00 34 00 20 00 09 00 28 00 1F 00 1C 00 06 00 ....4. ...(.......
00000036 00 00 34 00 00 00 34 80 04 08 34 80 04 08 20 01 00 00 ..4...4...4... ...
00000048 20 01 00 00 05 00 00 00 04 00 00 00 03 00 00 00 54 01  ...............T.
0000005a 00 00 54 81 04 08 54 81 04 08 13 00 00 00 13 00 00 00 ..T...T...........
0000006c 04 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 80 ..................
0000007e 04 08 00 80 04 08 2C 06 00 00 2C 06 00 00 05 00 00 00 ......,...,.......

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: 1.16843158668567E+305    Binary: 01111111 01000101 01
Show little endian decoding   Show unsigned as hexadecimal       ASCII Text: ELF
Offset: 0x0 / 0x1dd4       Selection: None       INS
oes not have a root element.
```

Explanation:-

The executable is divided into 3 sections.

1. From byte offset 0 to x = prefix
2. From x to y = P
3. From y to end = suffix

MD5(prefix || P || suffix) = MD5(prefix || Q || suffix)

The prefix is a multiple of 64 and a little above the byte offset of the first A. The byte offset is 1040 when we see continuous blocks of A.

Hence the byte offset is 4224 and the prefix is the first 4288 bytes. We use the following command of

head -c 4288 prog 1.out > prefix

With this command, we get two files with the same hash using the prefix file for md5collgen and they are p1 and p2. The command **md5collgen -p prefix -o a1 a2** results in files having a 10FF terminating byte offset. So, the byte after 10FF from the original is kept as the suffix. The command **tail -c +4416 prog1.out > suffix** is also used.

The individual files are concatenated. The following commands are used.

cat a1 suffix > file1

cat a2 suffix > file2

Finally, we can see that even though both the files differ, they have the same MD5 hashes. We use the following commands to demonstrate this.

Diff -q file1 file2

md5sum file1

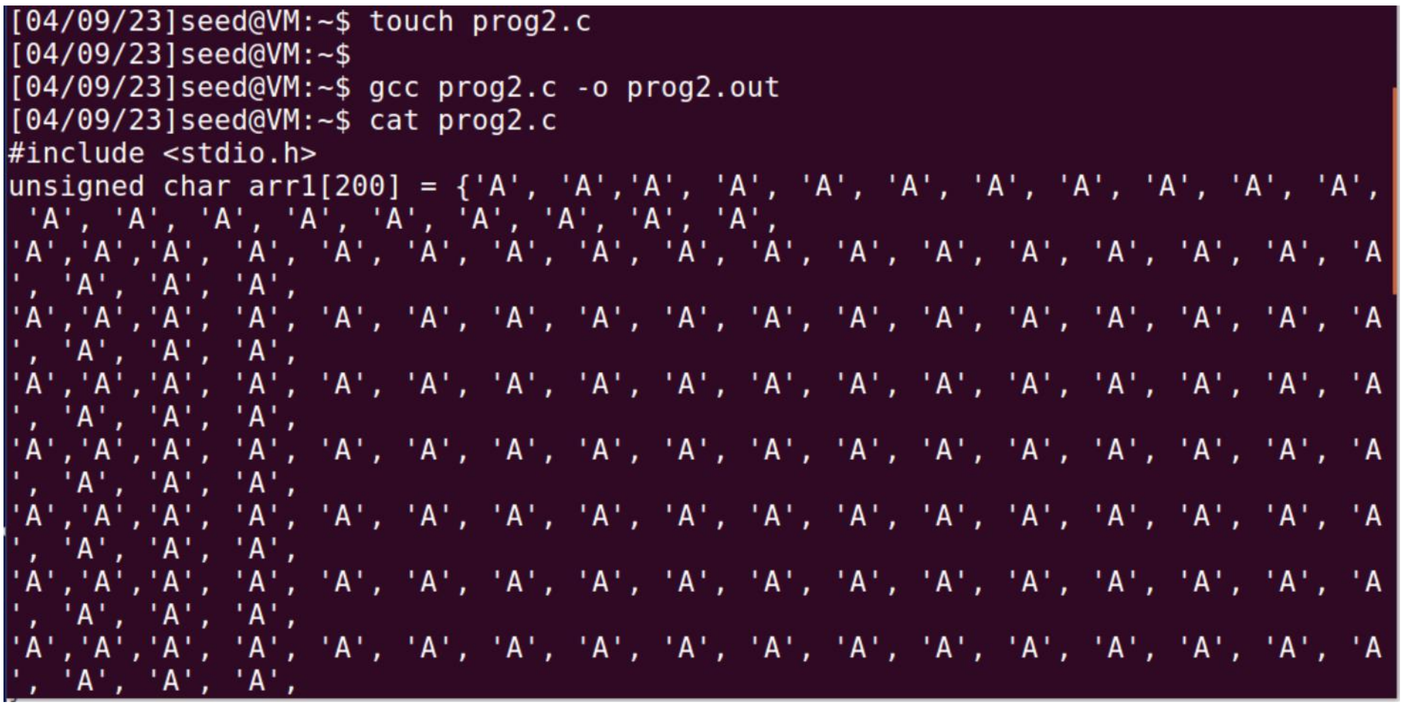
md5sum file2

Hence, we see that two different binaries are created but with the same hash value.

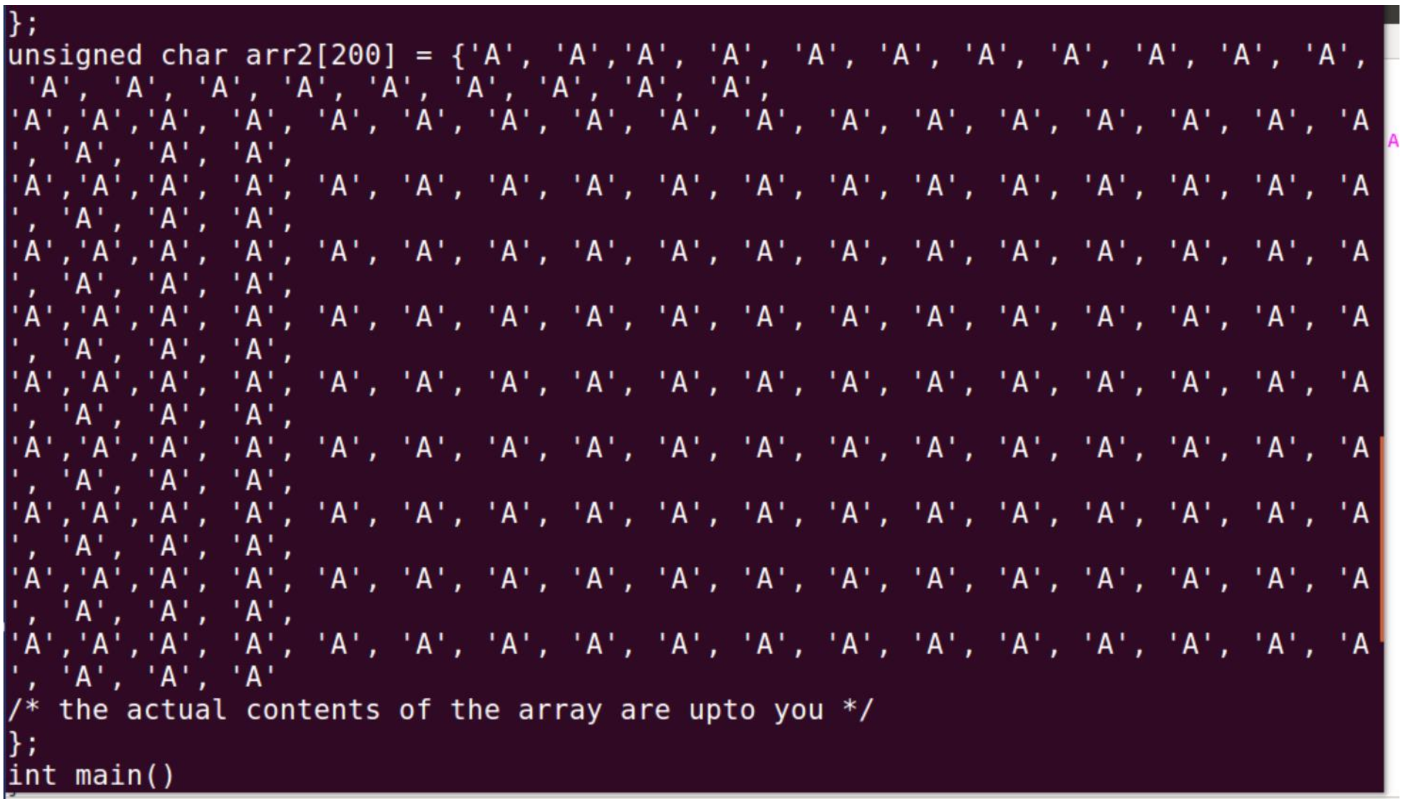
Task 4: Making the Two Programs Behave Differently

Below is the program in C

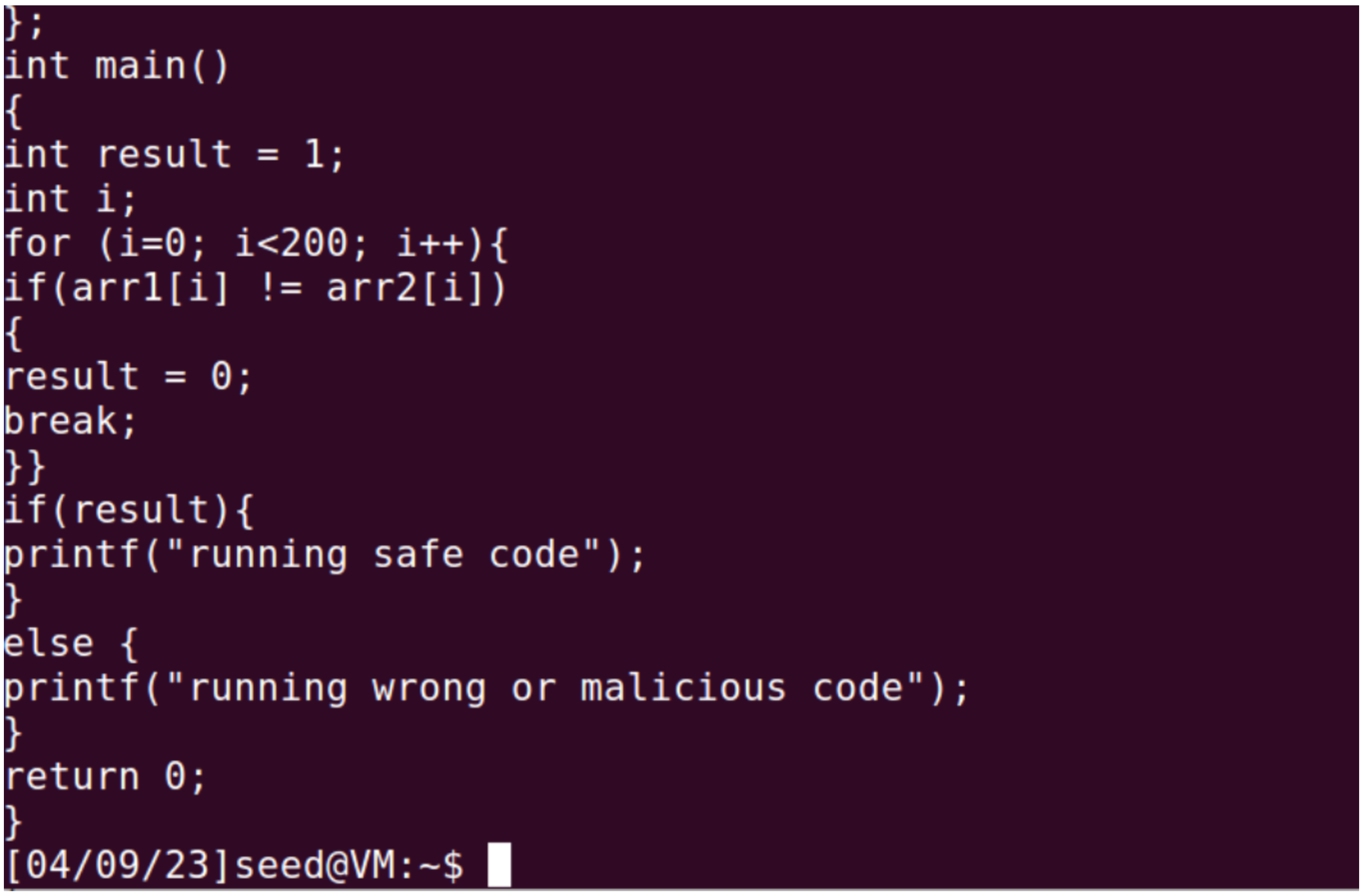
```
#include unsigned char arr1[200] = {'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'A', 'A','A', 'A', 'A', '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', 'A', 'A', 'A', 'A',
'A', 'A', 'A', 'A', 'A', 'A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A','A', 'A', 'A',
'A', 'A', 'A', 'A', 'A', 'A', 'A'

/* The actual contents of this array are up to you */

};

unsigned char arr2[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', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'A', 'A', 'A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A','A', 'A', 'A', 'A', 'A', 'A',
'A', 'A', 'A', 'A'

/* The actual contents of this array are up to you */

};

int main()

{

 int result = 1; int i;

for(int i=0; i<200; i++){

if(arr1[i] != arr2[i])

{

result = 0;

break;

}}

if(result){

printf("running safe code");

}

else {

printf("running wrong or malicious code");

}
return 0;

}
```



```
[04/09/23]seed@VM:~$ touch prog2.c
[04/09/23]seed@VM:~$
[04/09/23]seed@VM:~$ gcc prog2.c -o prog2.out
[04/09/23]seed@VM:~$ cat prog2.c
#include <stdio.h>
unsigned char arr1[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',
```



```
};
unsigned char arr2[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',
'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A
', 'A', 'A', 'A',
'A','A','A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A
', 'A', 'A', 'A'
/* the actual contents of the array are upto you */
};
int main()
```



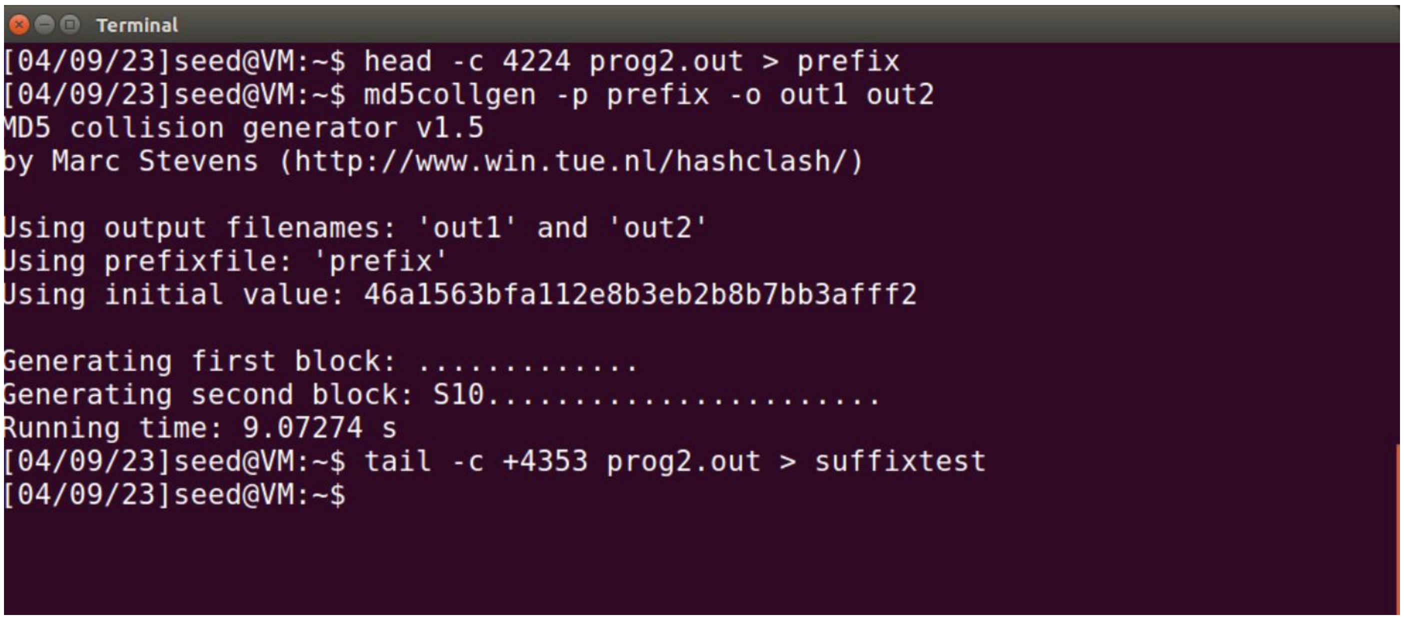
```
};
int main()
{
int result = 1;
int i;
for (i=0; i<200; i++){
if(arr1[i] != arr2[i])
{
result = 0;
break;
}}
if(result){
printf("running safe code");
}
else {
printf("running wrong or malicious code");
}
return 0;
}
[04/09/23]seed@VM:~$ 
```

After setting the prefix we generate 2 files from it which are out1 and out2 files. These files have all except the last 8 elements of the first array. We then add all the bytes after the 4352nd byte in prog2.out in the suffix. We use the following commands for this.

head -c 4224 prog2.out > prefix

md5collgen -p prefix -o out1 out2

tail -c +4353 task4.out > suffixtest



```
[04/09/23]seed@VM:~$ head -c 4224 prog2.out > prefix
[04/09/23]seed@VM:~$ md5collgen -p prefix -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix'
Using initial value: 46a1563bfa112e8b3eb2b8b7bb3afff2

Generating first block: ............
Generating second block: S10......................
Running time: 9.07274 s
[04/09/23]seed@VM:~$ tail -c +4353 prog2.out > suffixtest
[04/09/23]seed@VM:~$
```

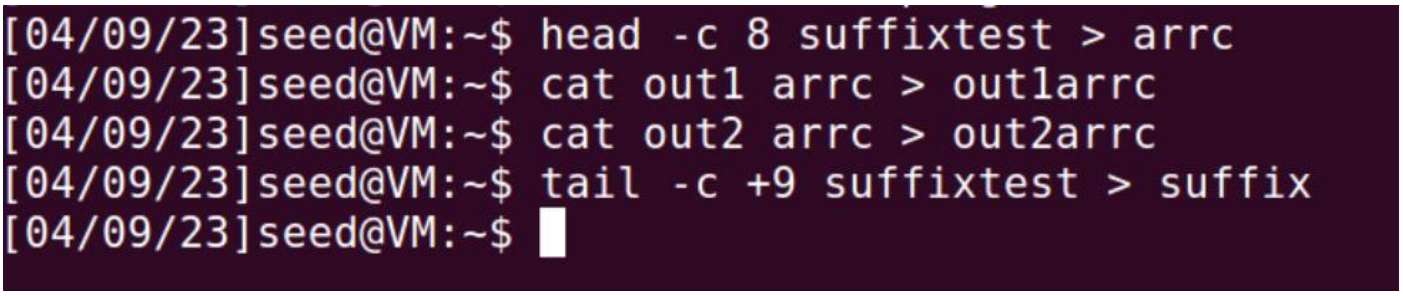
After this, we add the first eight bytes of suffixtest to both out1 and out2, which gives files out1arrc and out2arrc. After that, we create the suffix file, which contains all bytes after the eighth byte in suffixtest. The following commands are used in this part.

head -c 8 suffixtest > arrc

cat out1 arrc > out1arrc

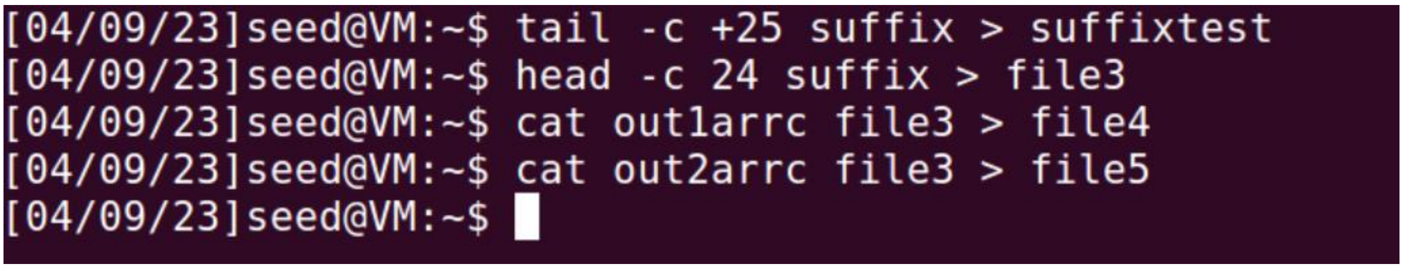
cat out2 arrc > out2arrc

tail -c +9 suffixtest > suffix



```
[04/09/23]seed@VM:~$ head -c 8 suffixtest > arrc
[04/09/23]seed@VM:~$ cat out1 arrc > out1arrc
[04/09/23]seed@VM:~$ cat out2 arrc > out2arrc
[04/09/23]seed@VM:~$ tail -c +9 suffixtest > suffix
[04/09/23]seed@VM:~$ 
```

We take the bytes between the end of the first array and the beginning of the second array and create a file file3. We store the bytes starting with the 2nd array in suffix to suffixtest. We then add these bytes to out1arrc and out2arrc which gives file4 and file5 respectively.



```
[04/09/23]seed@VM:~$ tail -c +25 suffix > suffixtest
[04/09/23]seed@VM:~$ head -c 24 suffix > file3
[04/09/23]seed@VM:~$ cat out1arrc file3 > file4
[04/09/23]seed@VM:~$ cat out2arrc file3 > file5
[04/09/23]seed@VM:~$ 
```

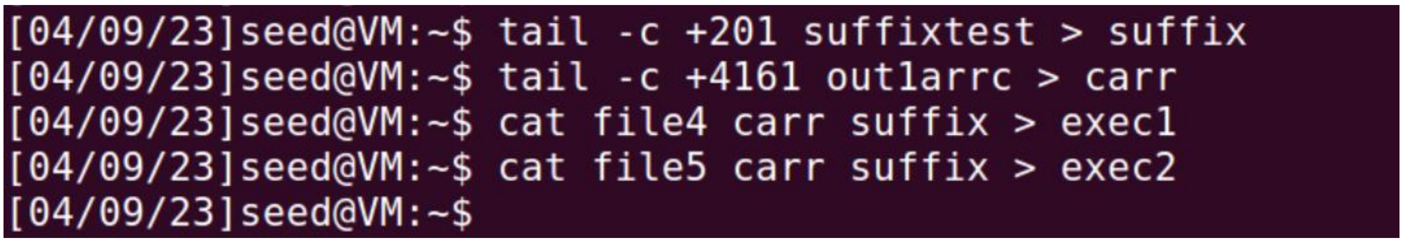
The two files are two separate parts of the program. The program is successful if one of the files prints "Running safe code!" while the other prints "Running malicious code!!!". To generate the second array, the contents of the first array need to be the same as one of the generated arrays. So, we put the bytes after the second array in suffixtest to suffix. The we copy the first array from out1arrc to carr. The file carr can be appended to file4 and file5 along with suffix which gives the final executables exec1 and exec2. The following commands are used.

tail -c +201 suffixtest > suffix

tail -c +4161 out1arrc > carr

cat file4 carr suffix > exec1

cat file5 carr suffix > exec2



```
[04/09/23]seed@VM:~$ tail -c +201 suffixtest > suffix
[04/09/23]seed@VM:~$ tail -c +4161 out1arrc > carr
[04/09/23]seed@VM:~$ cat file4 carr suffix > exec1
[04/09/23]seed@VM:~$ cat file5 carr suffix > exec2
[04/09/23]seed@VM:~$
```

Lastly, we calculate the md5sum and make both files executable. The following commands are used.

md5sum exec1

md5sum exec2

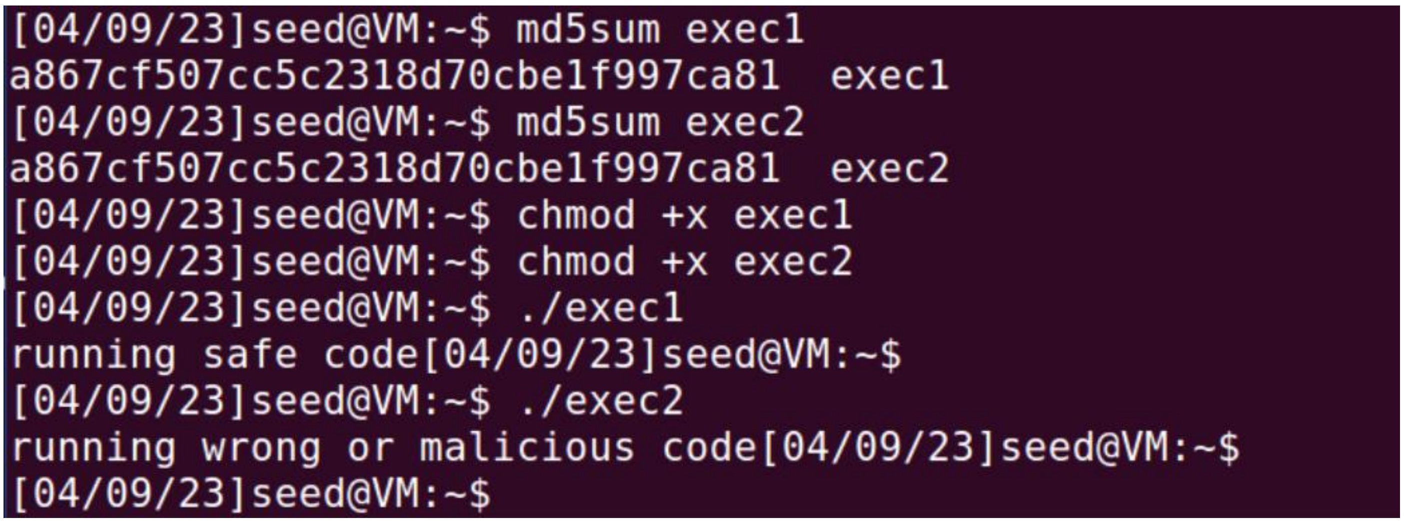
chmod +x exec1

chmod +x exec2

./exec1

./exec2

This is the way in which we exploit md5 vulnerability.



```
[04/09/23]seed@VM:~$ md5sum exec1
a867cf507cc5c2318d70cbe1f997ca81  exec1
[04/09/23]seed@VM:~$ md5sum exec2
a867cf507cc5c2318d70cbe1f997ca81  exec2
[04/09/23]seed@VM:~$ chmod +x exec1
[04/09/23]seed@VM:~$ chmod +x exec2
[04/09/23]seed@VM:~$ ./exec1
running safe code[04/09/23]seed@VM:~$
[04/09/23]seed@VM:~$ ./exec2
running wrong or malicious code[04/09/23]seed@VM:~$
[04/09/23]seed@VM:~$
```

Submitted By: -

Aastha Dhir

CWID-A20468022

adhir2@hawk.iit.edu