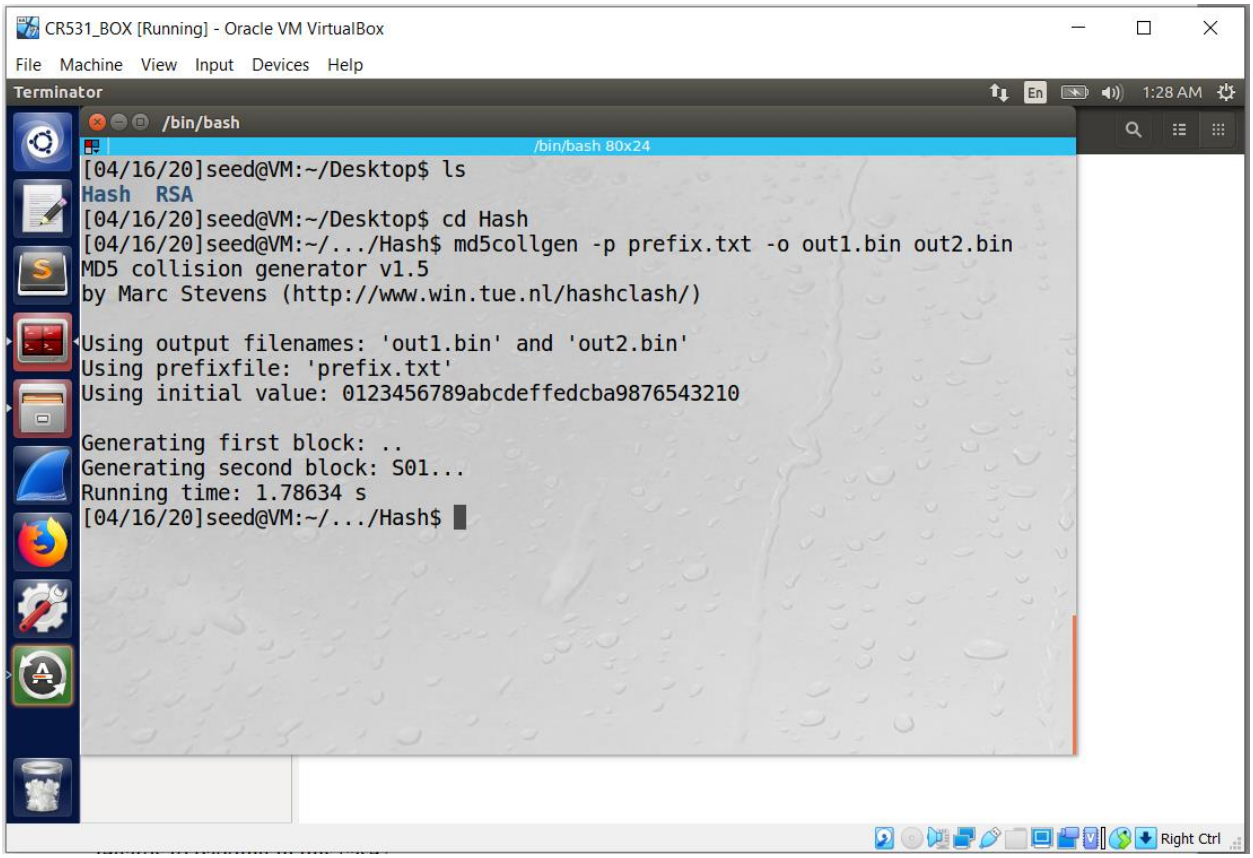
CS-767 FOUNDATIONS NETWORK SECURITY

HASH LAB

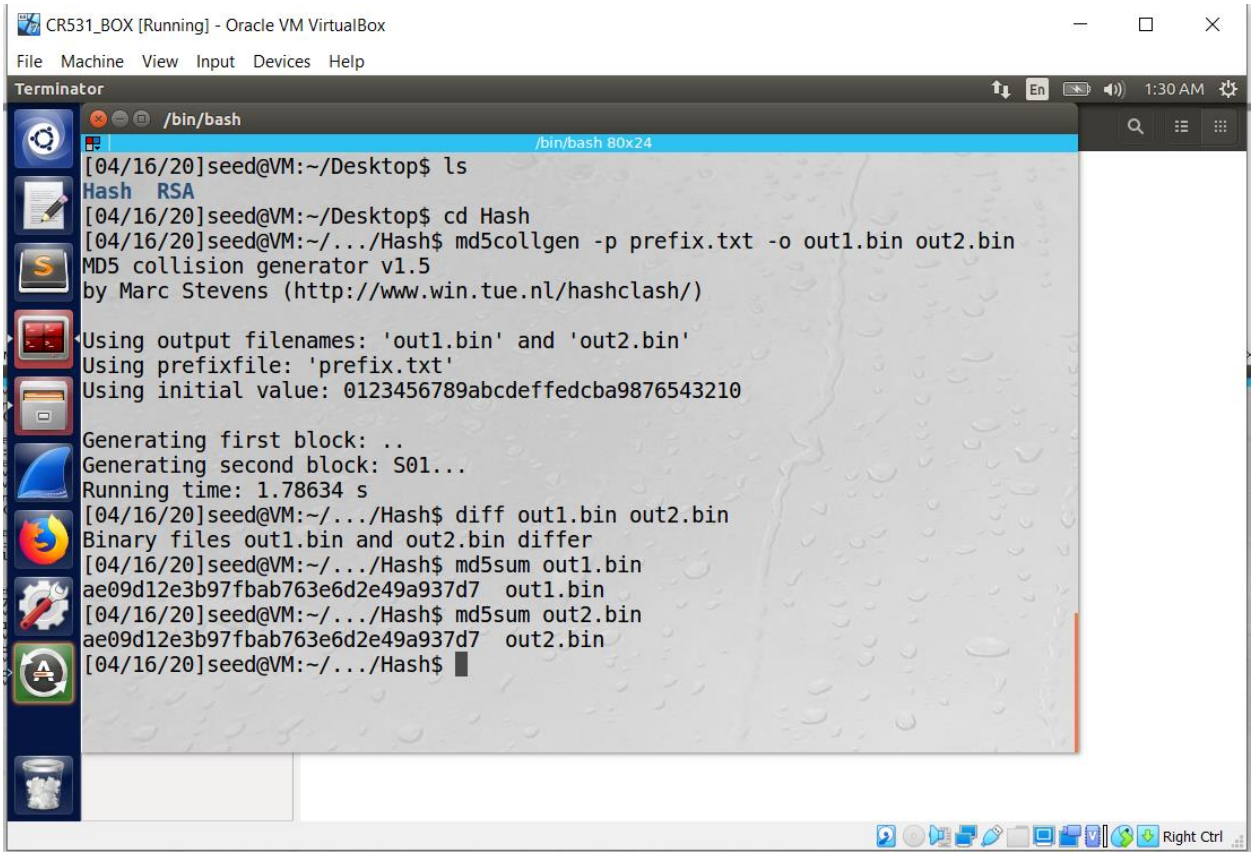
Chakradhar Reddy Donuri

E949F496

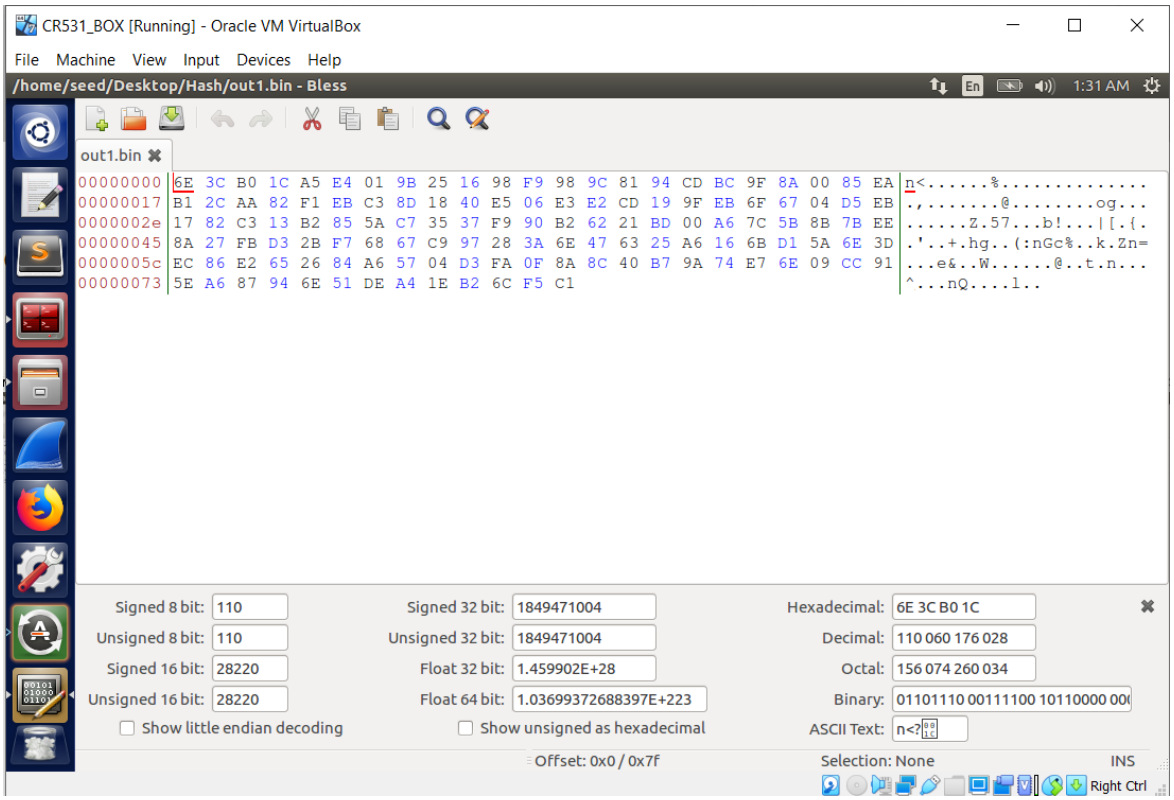
Task-1: The following command generates two output files, out1.bin and out2.bin, for a given a prefix file prefix.txt: $ md5collgen -p prefix.txt -o out1.bin out2.bin



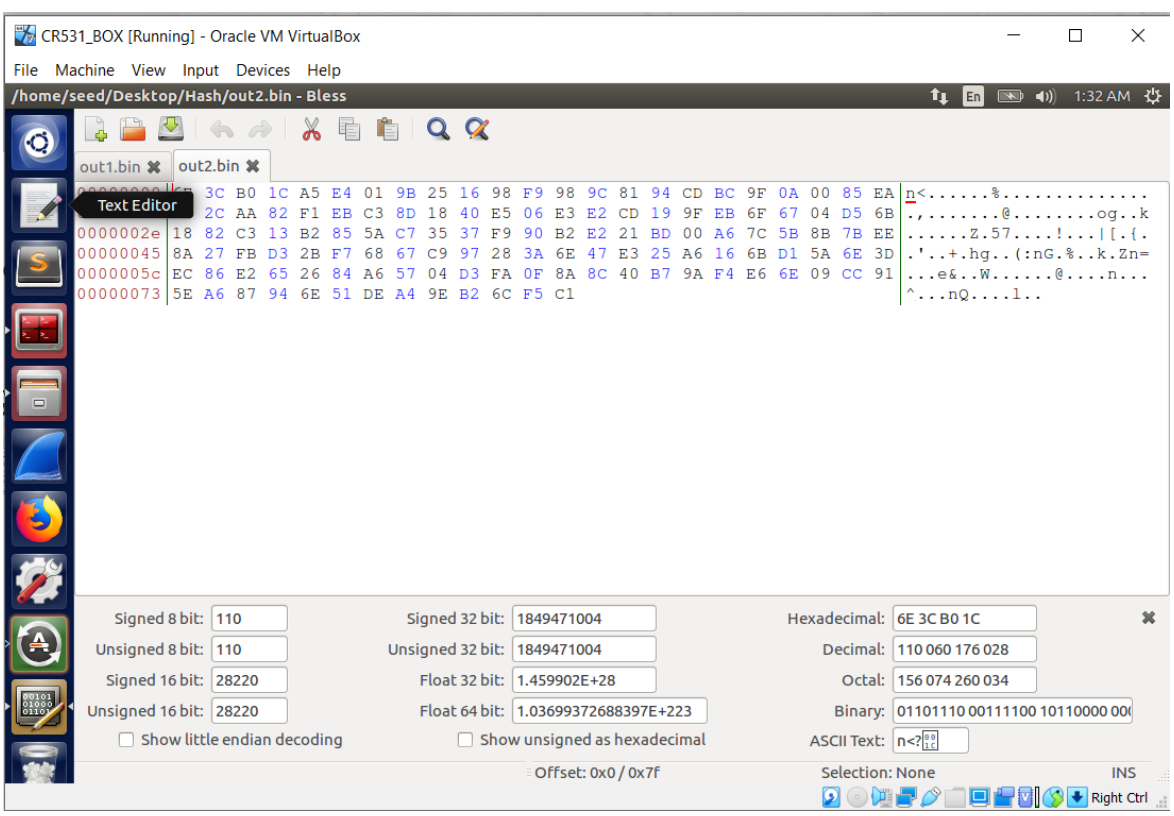
Now We can check whether the output files are distinct or not using the diff command. We can also use the md5sum command to check the MD5 hash of each output file



Out1.bin in Bless Editor



Out2.bin in Bless editor



1. Suppose that you use an arbitrary prefix file that has a byte length that is not a multiple of 64 as your input to md5collgen. What will md5collgen do in terms of padding?
Ans: Extra zeros are padded

```
out1.bin ✖   out2.bin ✖
00000000 6E 3C B0 1C A5 E4 01 9B 25 16 98 F9 98 9C 81 94 CD BC 9F 8A 00 85 EA B1 2C AA 82 F1 EB C3
0000001e 8D 18 40 E5 06 E3 E2 CD 19 9F EB 6F 67 04 D5 EB 17 82 C3 13 B2 85 5A C7 35 37 F9 90 B2 62
0000003c 21 BD 00 A6 7C 5B 8B 7B EE 8A 27 FB D3 2B F7 68 67 C9 97 28 3A 6E 47 63 25 A6 16 6B D1 5A
0000005a 6E 3D EC 86 E2 65 26 84 A6 57 04 D3 FA 0F 8A 8C 40 B7 9A 74 E7 6E 09 CC 91 5E A6 87 94 6E
00000078 51 DE A4 1E B2 6C F5 C1
```

2. Suppose that you a prefix file that is 64 bytes long as input to md5collgen again. What happens in regards to padding in this case?
Ans: If it is exact 64 bytes then no extra zeros are padded.

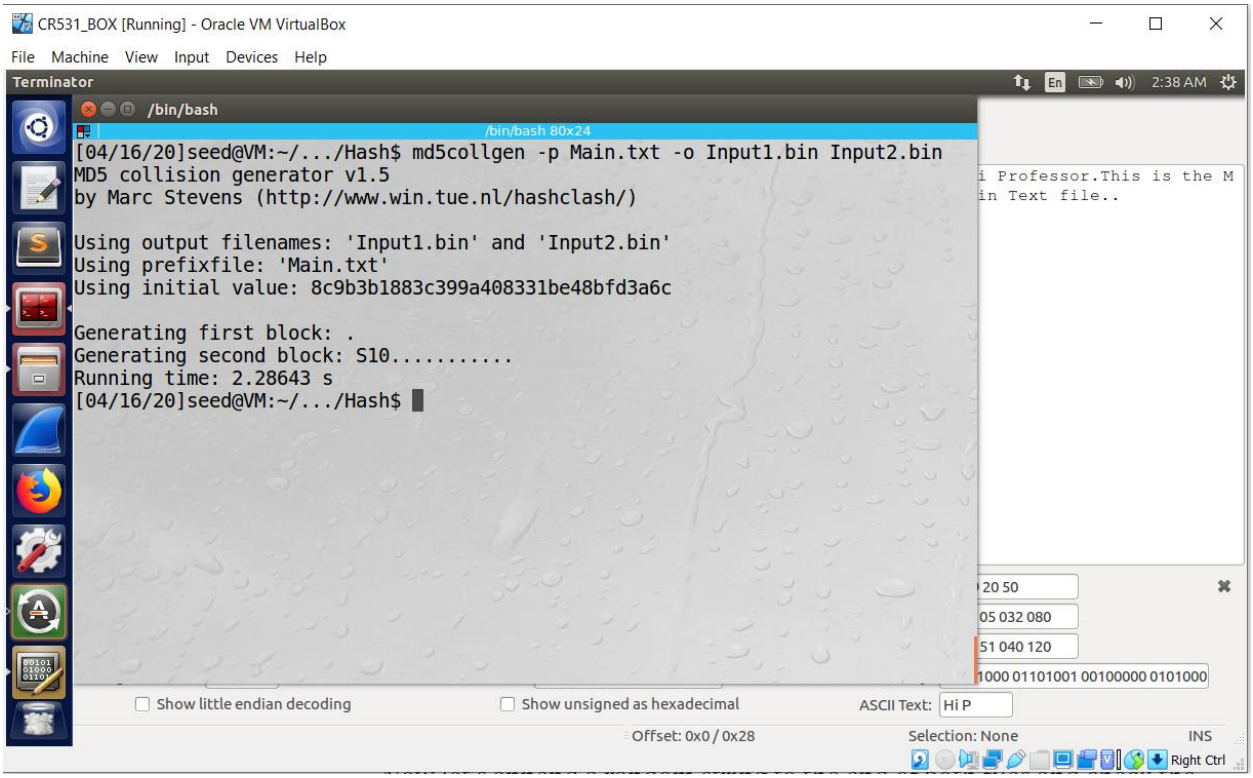
3. Create an arbitrary prefix file and generate two binaries with the same MD5 sum as explained above. The files generated by md5collgen are different. Which bytes are different and what are their values? You can use bless and/or cmp to find the answer.
Ans: From the Bless editor screenshots attached above it is observed that
Bytes that are different (20,46,47,60,84,110,111,124)
Values at those positions in out1.bin are (8A, EB, 17, 62, 63, 74, E7, 1E)
Values at those positions in out2.bin are (0A, 6B, 18, E2, E3, F4, E6, 9E)

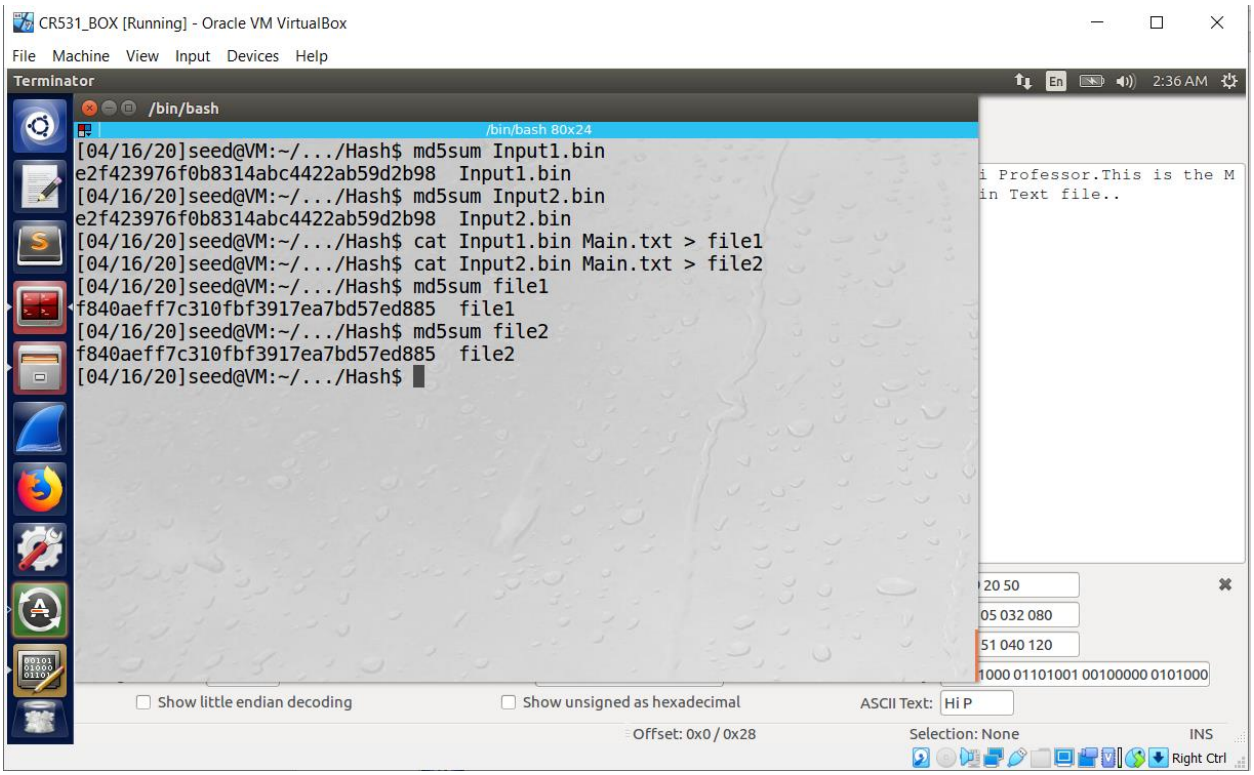
4. The output files are different but they have the same MD5 sum. Identify the cryptographic hash property that is not met by the MD5 hash.
Ans: Two different messages or file with same MD5 sum This property is sometimes referred to as strong collision resistance.

5. Identify an attack that can be launched due to the vulnerability that you identified above
Ans: Collision attack on a cryptographic hash that tries to find two inputs files or messages producing the same hash value, i.e. a hash collision.

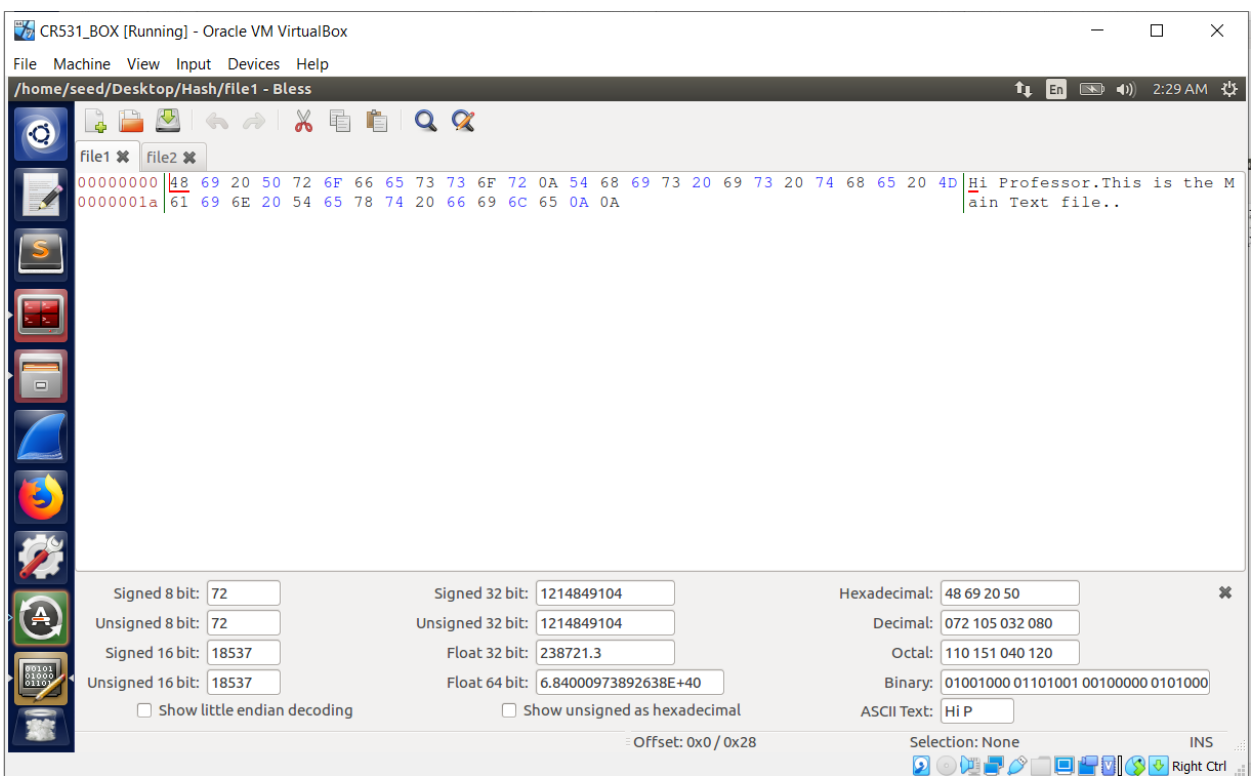
Task-2: Adding a suffix to two files that have the same hash value

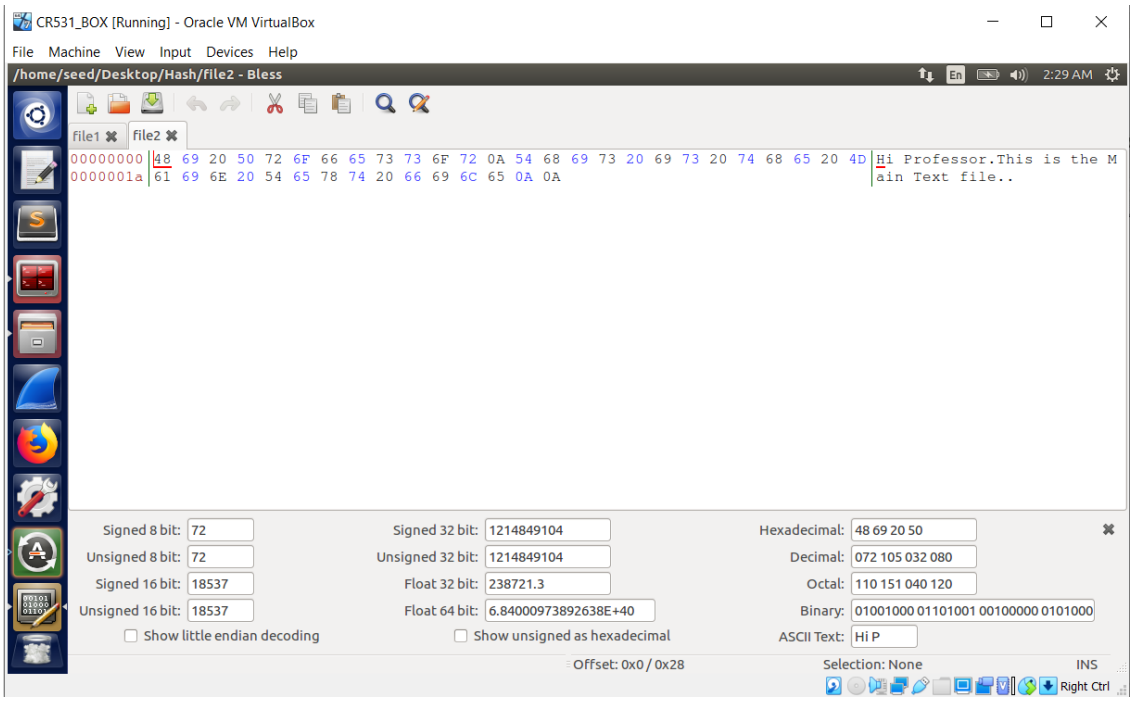
Consider 2 input files names Input1.bin and Input2.bin and a file that is used to concatenate is Main.txt





File1 and File2 In Bless Editor





From the above experiment, we can observe that this property holds for MD5 i.e. Input files have same hash value and adding the same Main.txt file to each Input file resulted in two outputs that have same hash value.

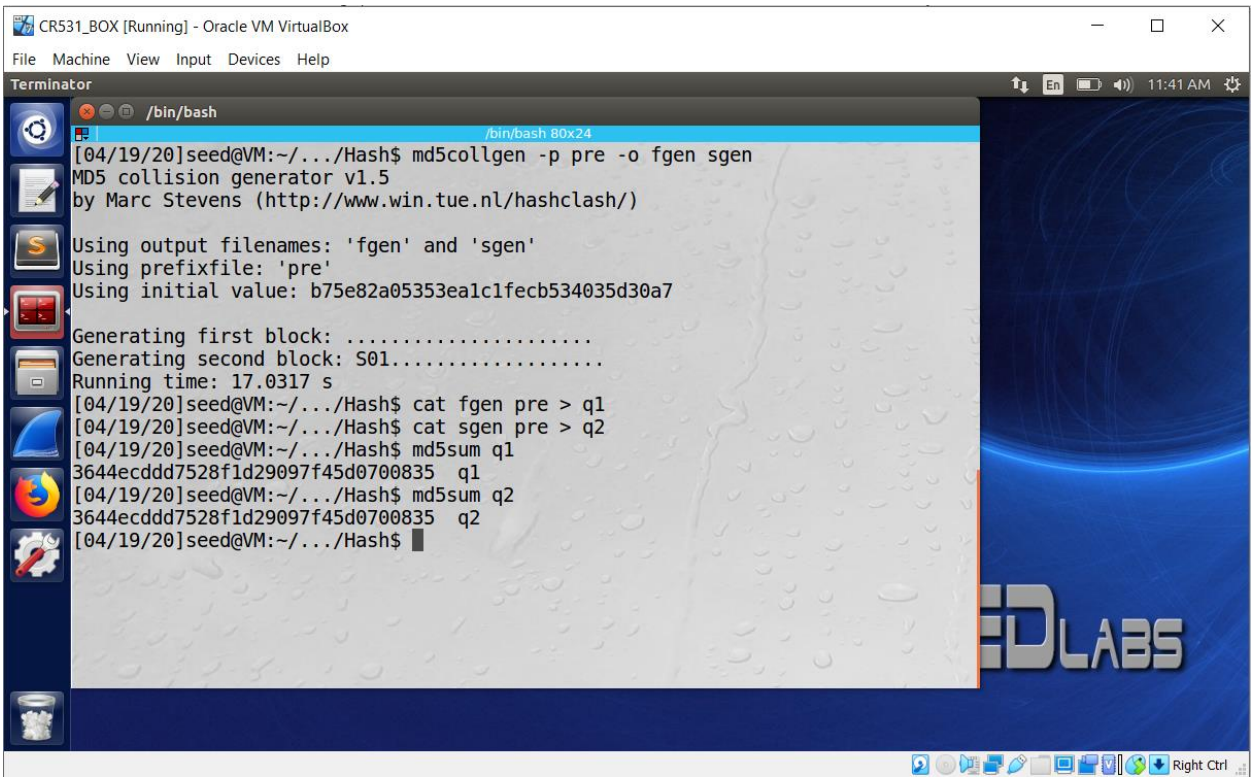
Task-3: Generating Two Executable Files with the Same MD5 Hash

1) The ending byte position of your prefix and the starting byte position of your suffix in the original C program?

Ans) Now, we head -c 64 a.out > pre and tail -c +4320 program > suf

Ending byte of pre in bless : 1040
Starting byte of suf in bless : 10e0

2) A screen shot showing that the output of md5sum for both of your programs (i.e., the ones that are built with the two outputs from md5collgen) is the same.
Ans)



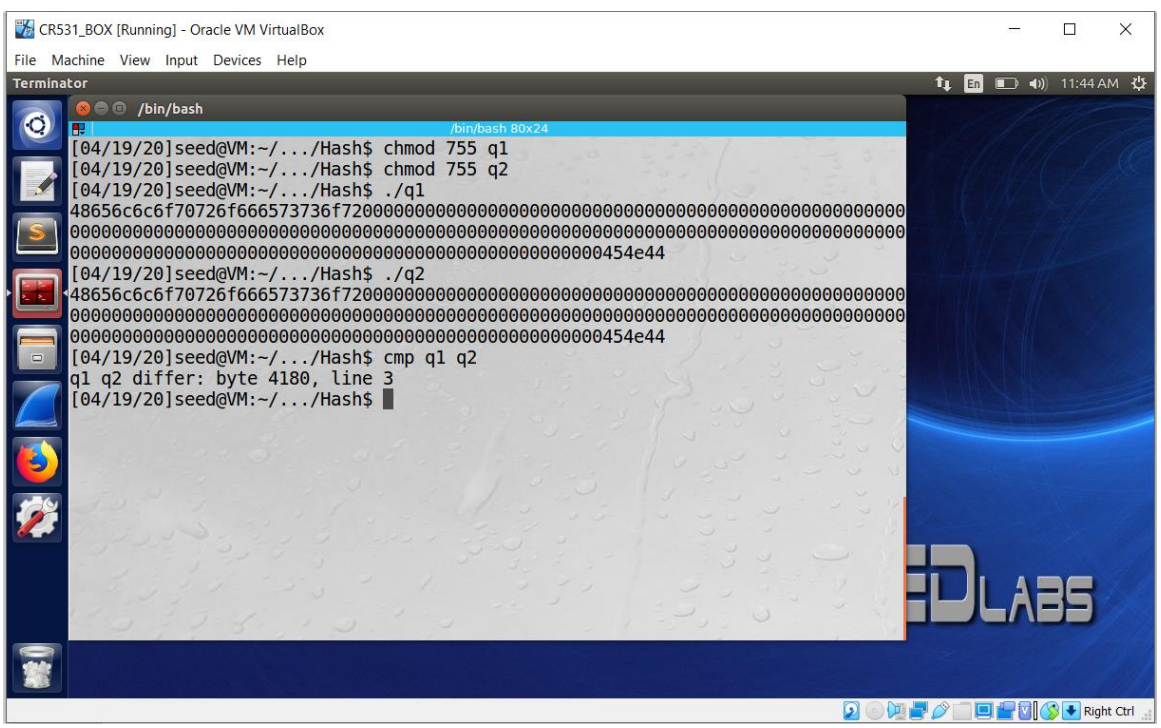
```
[04/19/20]seed@VM:~/.../Hash$ md5collgen -p pre -o fgen sgen
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'fgen' and 'sgen'
Using prefixfile: 'pre'
Using initial value: b75e82a05353ea1c1fecb534035d30a7

Generating first block: ......................
Generating second block: S01..................
Running time: 17.0317 s
[04/19/20]seed@VM:~/.../Hash$ cat fgen pre > q1
[04/19/20]seed@VM:~/.../Hash$ cat sgen pre > q2
[04/19/20]seed@VM:~/.../Hash$ md5sum q1
3644ecddd7528f1d29097f45d0700835  q1
[04/19/20]seed@VM:~/.../Hash$ md5sum q2
3644ecddd7528f1d29097f45d0700835  q2
[04/19/20]seed@VM:~/.../Hash$ 
```

Md5sum for both the programs is same.

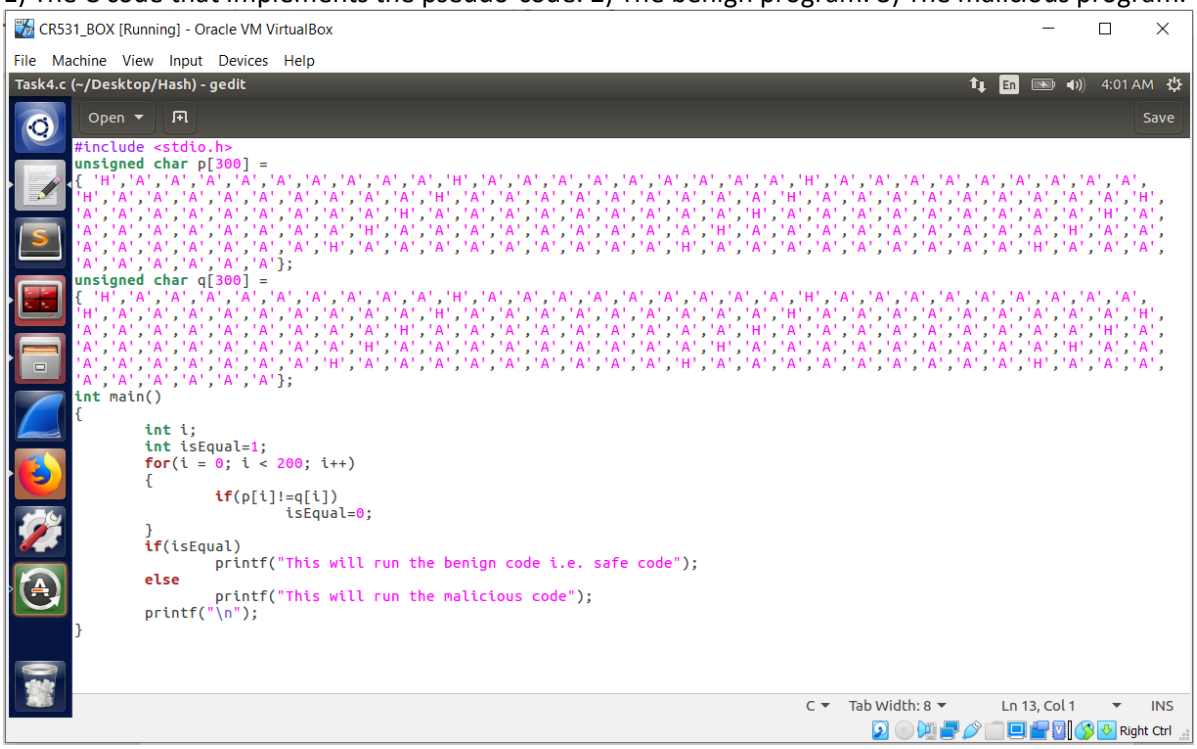
3 and 4) The output of cmp for both of your programs showing that they are different and The output of your programs showing that they still run and print the array to the screen.



```
[04/19/20]seed@VM:~/.../Hash$ chmod 755 q1
[04/19/20]seed@VM:~/.../Hash$ chmod 755 q2
[04/19/20]seed@VM:~/.../Hash$ ./q1
48656c6c6f70726f666573736f720000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000454e44
[04/19/20]seed@VM:~/.../Hash$ ./q2
48656c6c6f70726f666573736f720000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000454e44
[04/19/20]seed@VM:~/.../Hash$ cmp q1 q2
q1 q2 differ: byte 4180, line 3
[04/19/20]seed@VM:~/.../Hash$ 
```

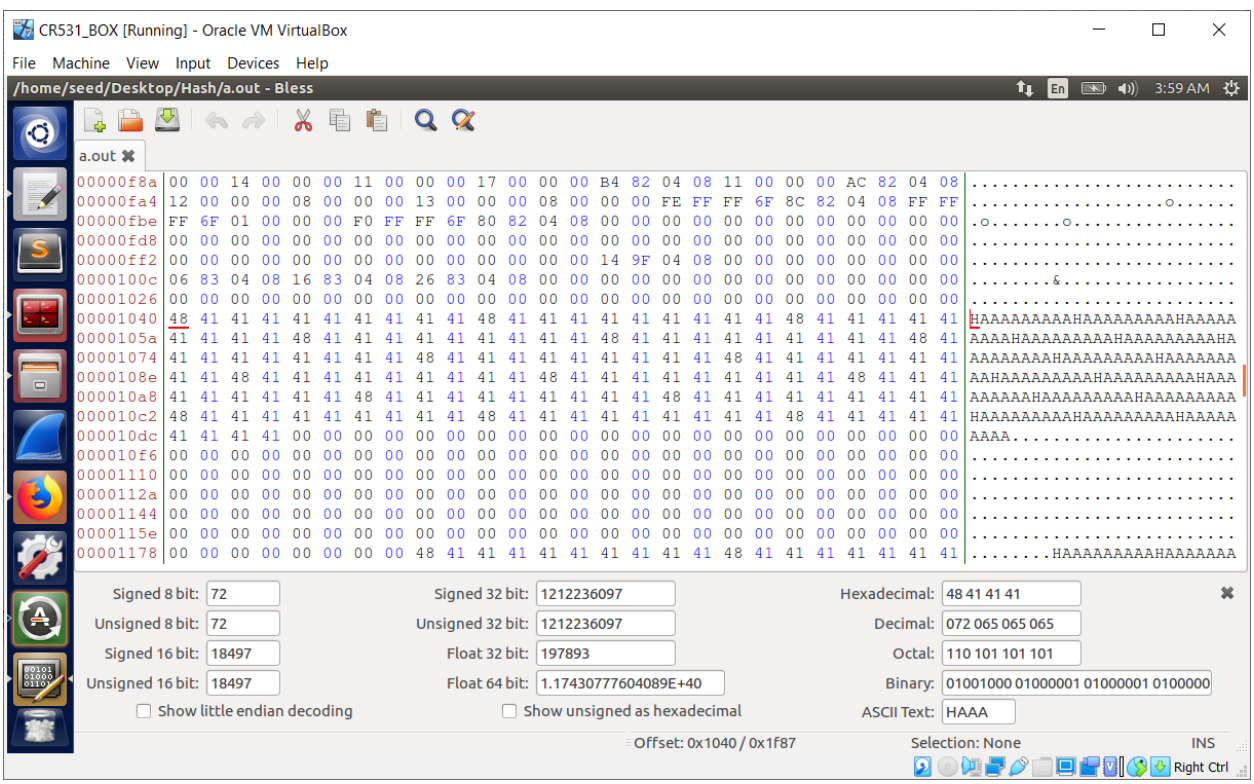
Task-4: Making the Two Programs Behave Differently

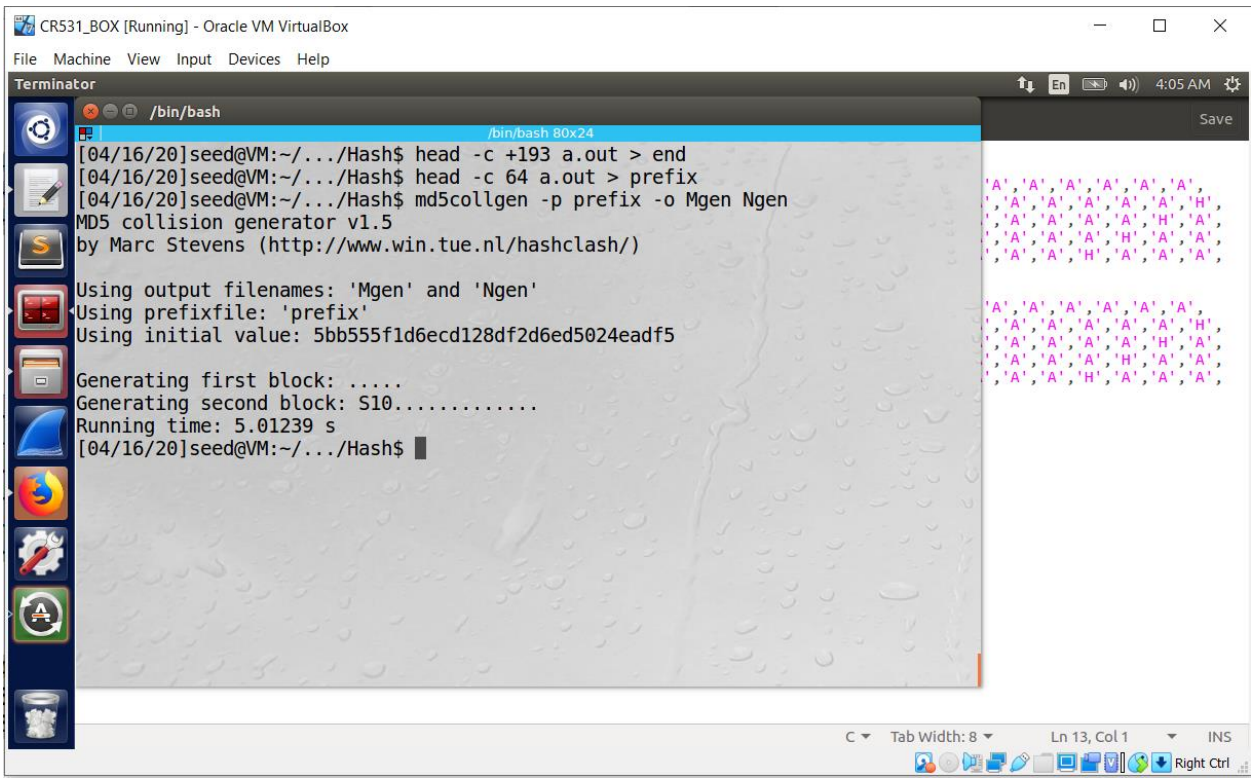
1) The C code that implements the pseudo-code. 2) The benign program. 3) The malicious program.



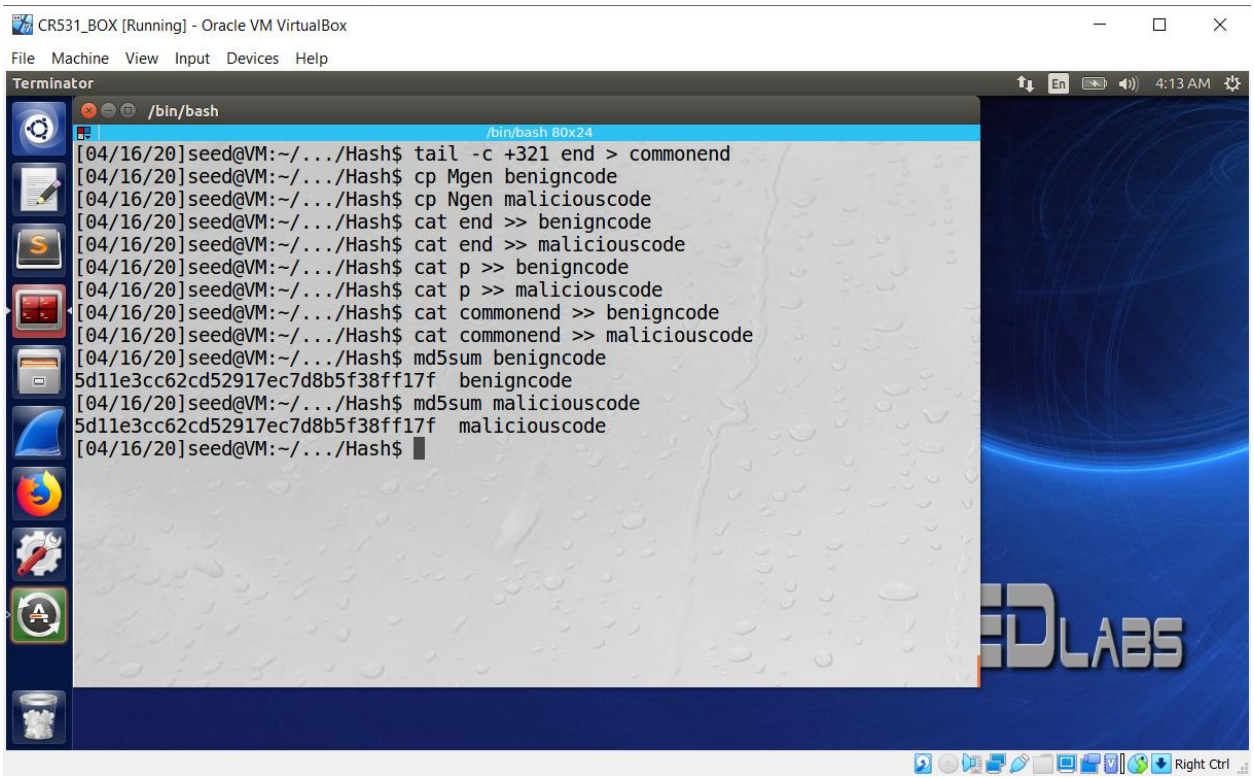

```
#include <stdio.h>
unsigned char p[300] =
{ 'H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A',
'H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H',
'A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A',
'A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A','A',
'A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A','A','A',
'A','A','A','A','A','A'};
unsigned char q[300] =
{ 'H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A',
'H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H',
'A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A',
'A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A','A',
'A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A','H','A','A','A',
'A','A','A','A','A','A'};
int main()
{
        int i;
        int isEqual=1;
        for(i = 0; i < 200; i++)
        {
                if(p[i]!=q[i])
                        isEqual=0;
        }
        if(isEqual)
                printf("This will run the benign code i.e. safe code");
        else
                printf("This will run the malicious code");
        printf("\n");
}
```

Output of Task4.c in Bless editor



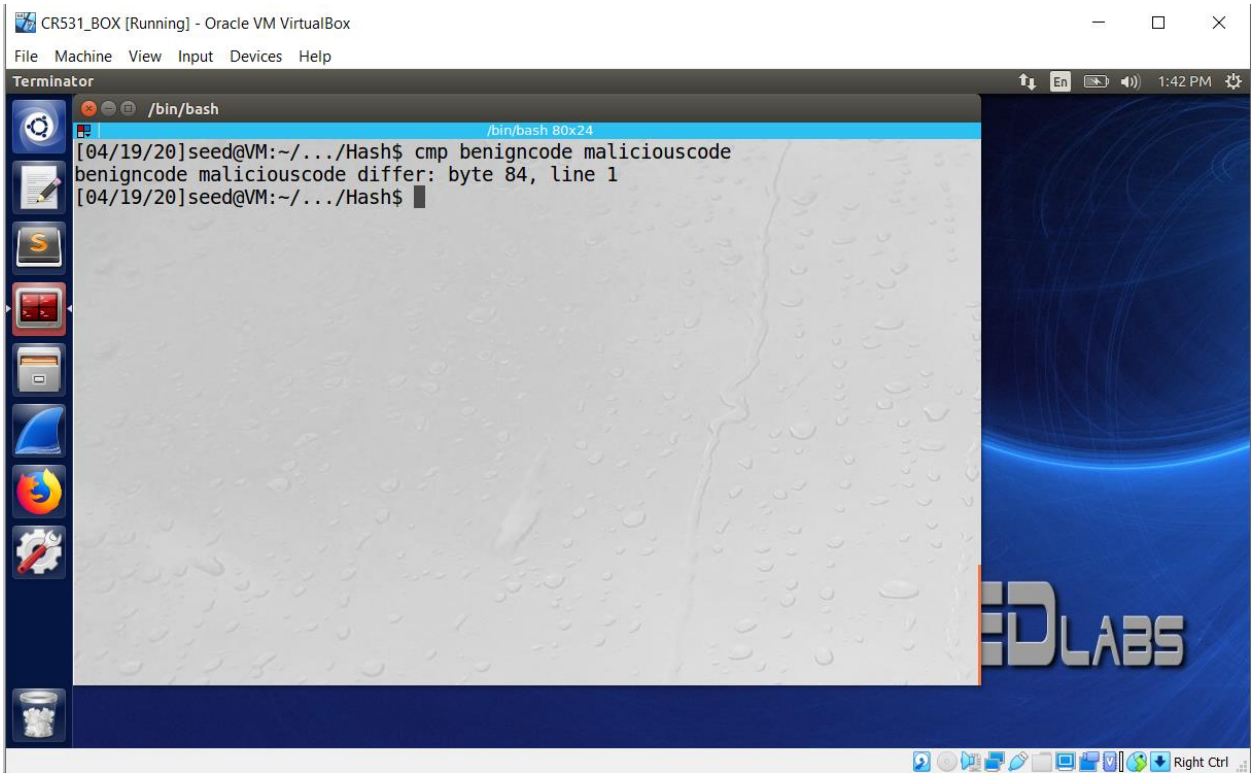


4) Screen shot of the output of md5sum for the binaries of the benign and malicious programs

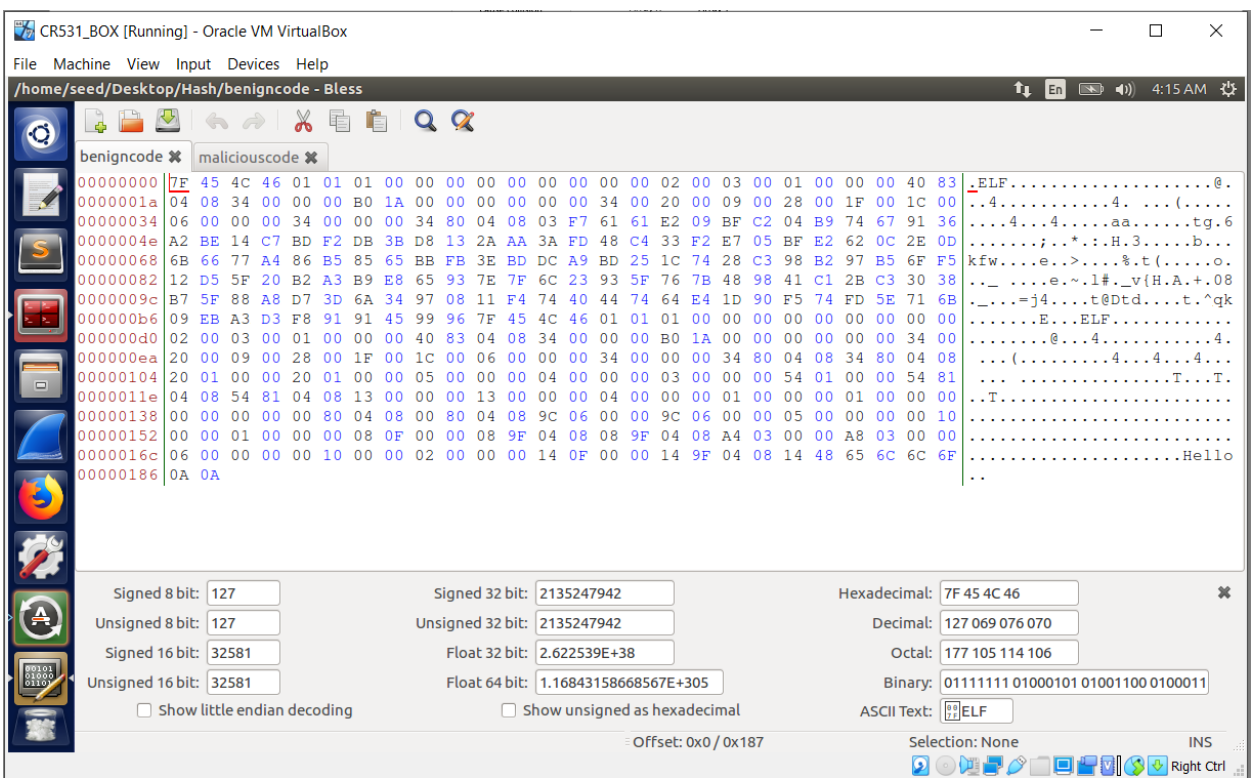


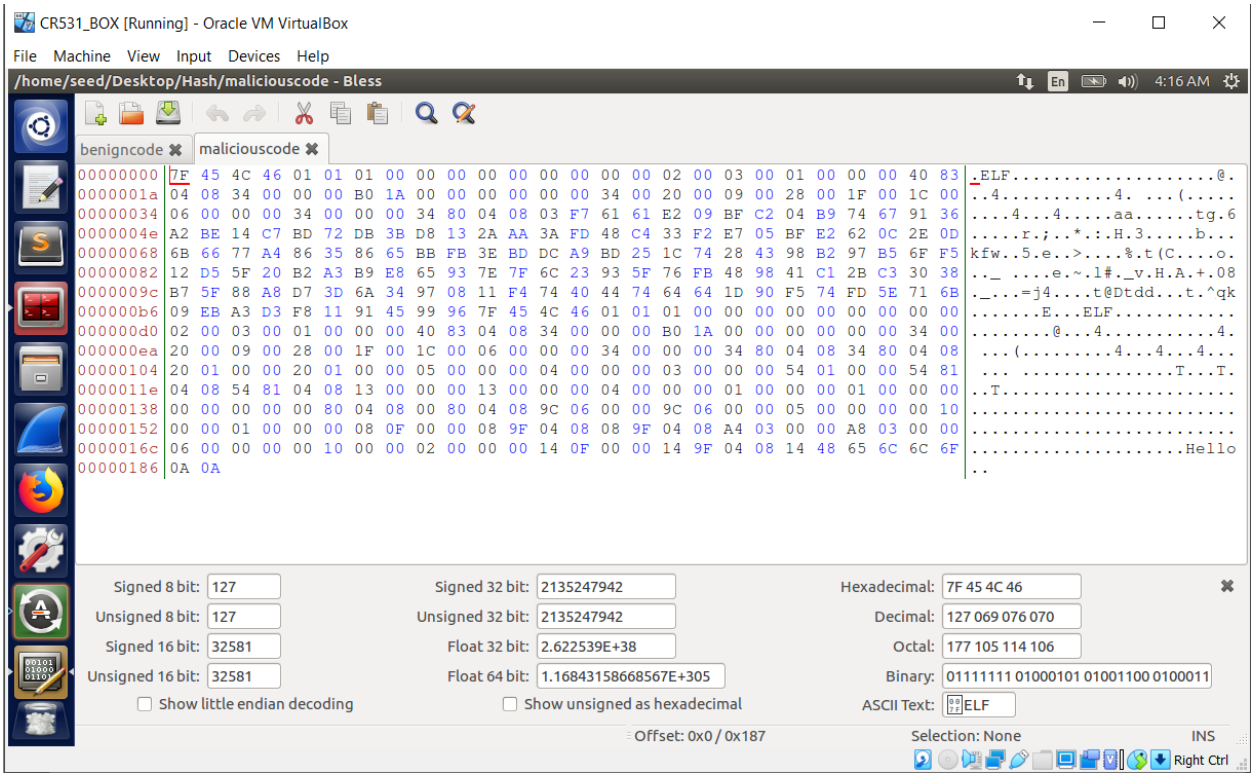
They have same Hash values.

Screen shot of the output of cmp to compare the contents of the malicious and benign binaries



Benigncode and maliciouscode in Bless editor





Hence, benigncode and malicious code are different from above screenshots.