SEED LAB: MD5 Collision Attack

Contents

Task 1

- I created a file named prefix in the folder where I placed md5collgen to generate two output files with 2 different hashes. Moreover, I have added string “seed lab task in progress” in the *prefix.txt* file to begin.

```
seed@VM: ~/.../SEED
[10/12/22]seed@VM:~$ cd Desktop
[10/12/22]seed@VM:~/Desktop$ cd SEED
[10/12/22]seed@VM:~/.../SEED$ touch prefix.txt
[10/12/22]seed@VM:~/.../SEED$ ls *.txt
prefix.txt
[10/12/22]seed@VM:~/.../SEED$ echo "seed lab task in progress" >>
refix.txt
[10/12/22]seed@VM:~/.../SEED$ cat prefix.txt
seed lab task in progress
```

- I used md5collgen to generate two different outputs.

```
[10/12/22]seed@VM:~/.../SEED$ md5collgen -p prefix.txt -o out1.bin
output2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'output2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 2dca35cef8ab6762cff635f0fd26a40d

Generating first block: ........................................
Generating second block: S00.............
Running time: 38.1563 s
[10/12/22]seed@VM:~/.../SEED$ ls
md5collgen  out1.bin  out2.bin  output2.bin  prefix.txt  Walter
[10/12/22]seed@VM:~/.../SEED$
```

- Here, I used the command given in the manual to check if the binaries of the output files differ and yes they do.

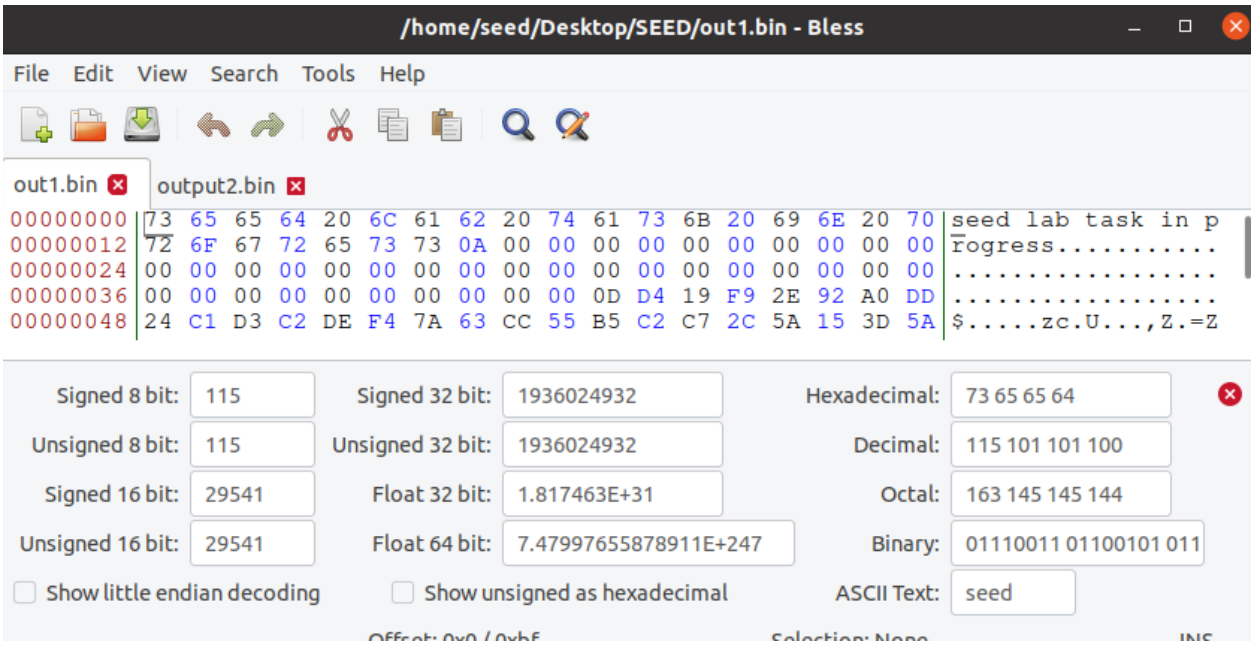
```
[10/12/22]seed@VM:~/.../SEED$ diff out1.bin output2.bin
Binary files out1.bin and output2.bin differ
[10/12/22]seed@VM:~/.../SEED$
```

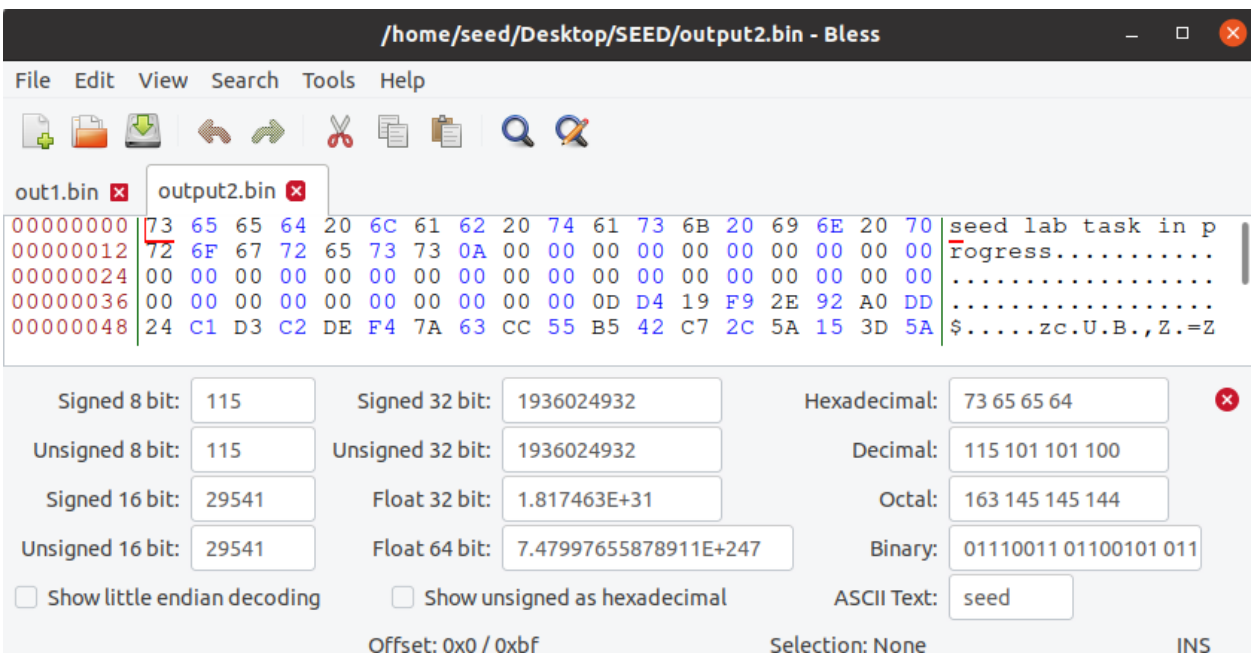
```
[10/12/22]seed@VM:~/.../SEED$ xxd out1.bin
00000000: 7365 6564 206c 6162 2074 6173 6b20 696e  seed lab task in
00000010: 2070 726f 6772 6573 730a 0000 0000 0000   progress.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0dd4 19f9 2e92 a0dd 24c1 d3c2 def4 7a63  ........$.....zc
00000050: cc55 b5c2 c72c 5a15 3d5a 4141 4fb2 a62a  .U...,Z.=ZAAO..*
00000060: dee5 fded bcb3 db38 4b42 bd9a dd8d f987  .......8KB......
00000070: bbc0 0e64 8a36 bd7d ef2d e6b9 16e3 3fa0  ...d.6.}.-....?.
00000080: ee0d 2a2e ce76 cdff b966 1689 635f eb46  ..*..v...f..c_.F
00000090: ea37 4aeb 2ebc c020 d9fb f3c0 5459 799e  .7J.... ....TYy.
000000a0: 8b99 5ebe 2936 9777 7d54 9d0f 7ce6 4ec8  ..^.)6.w}T..|.N.
000000b0: dcba efa5 70b7 32c9 a0e4 ed32 d1f2 3d31  ....p.2....2..=1
[10/12/22]seed@VM:~/.../SEED$ xxd output2.bin
00000000: 7365 6564 206c 6162 2074 6173 6b20 696e  seed lab task in
00000010: 2070 726f 6772 6573 730a 0000 0000 0000   progress.......
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0dd4 19f9 2e92 a0dd 24c1 d3c2 def4 7a63  ........$.....zc
00000050: cc55 b542 c72c 5a15 3d5a 4141 4fb2 a62a  .U.B.,Z.=ZAAO..*
00000060: dee5 fded bcb3 db38 4b42 bd9a dd0d fa87  .......8KB......
00000070: bbc0 0e64 8a36 bd7d ef2d e639 16e3 3fa0  ...d.6.}.-.9..?.
00000080: ee0d 2a2e ce76 cdff b966 1689 635f eb46  ..*..v...f..c_.F
00000090: ea37 4a6b 2ebc c020 d9fb f3c0 5459 799e  .7Jk... ....TYy.
000000a0: 8b99 5ebe 2936 9777 7d54 9d0f 7c66 4ec8  ..^.)6.w}T..|fN.
000000b0: dcba efa5 70b7 32c9 a0e4 edb2 d1f2 3d31  ....p.2.......=1
[10/12/22]seed@VM:~/.../SEED$
```

- Checking the Hash values of both output files to confirm the difference in files to view the output files which they evidently differ as shown in the screenshot below.

```
[10/12/22]seed@VM:~/.../SEED$ md5sum out1.bin
5491f17b160c0ee306bb808f947dc683  out1.bin
[10/12/22]seed@VM:~/.../SEED$ md5sum out2.bin
d41d8cd98f00b204e9800998ecf8427e  out2.bin
[10/12/22]seed@VM:~/.../SEED$
```

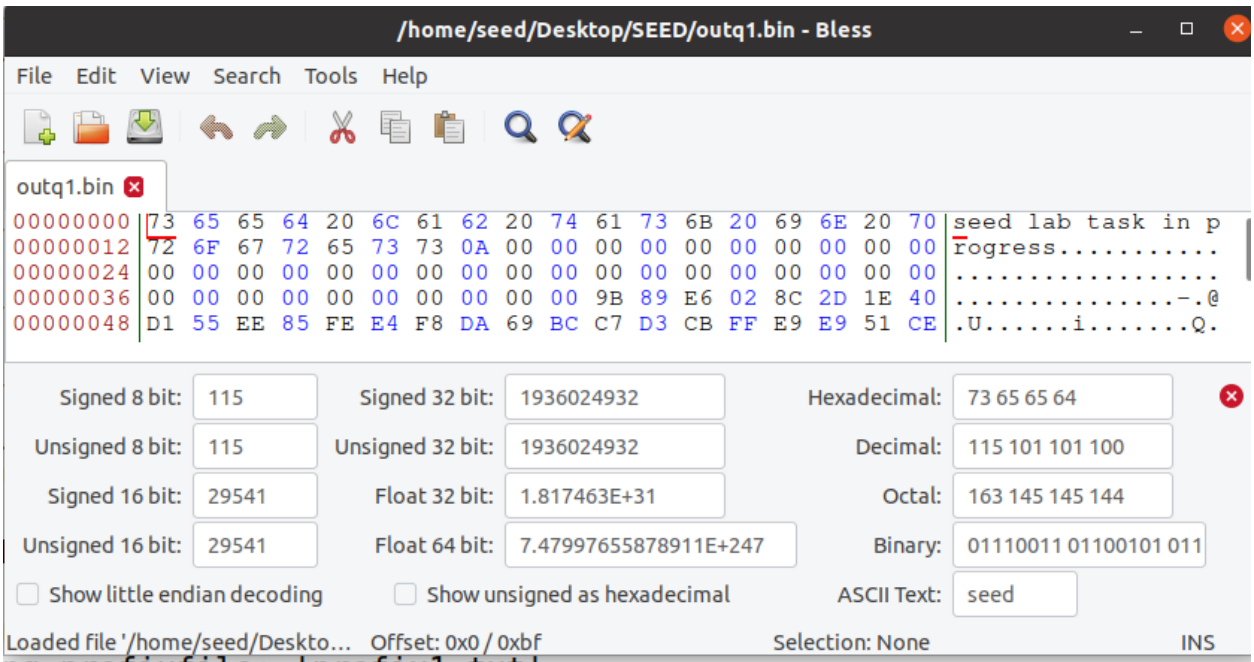
- Now I opened both the files *out1.bin* and *ouput2.bin* to check their hex values to verify the difference.



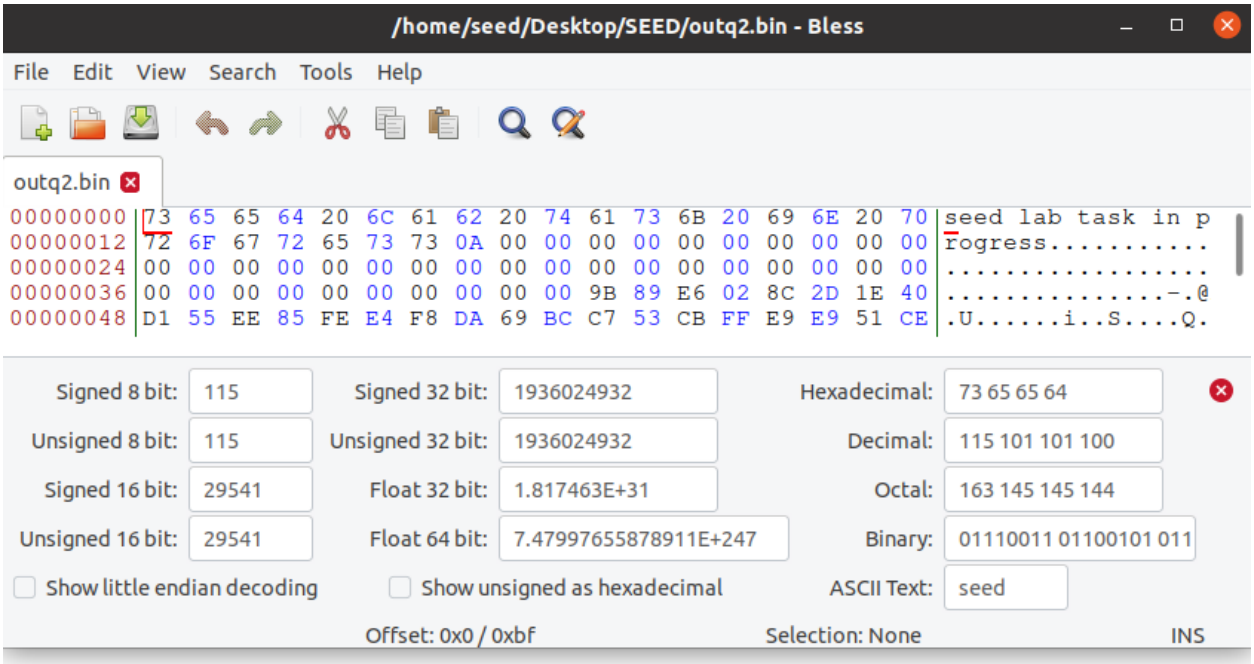


Question 1

- I created a new file with string “seed lab task in progress” named as prefix1.txt. I used md5collgen to generate different output files in order to answer the questions in task1.

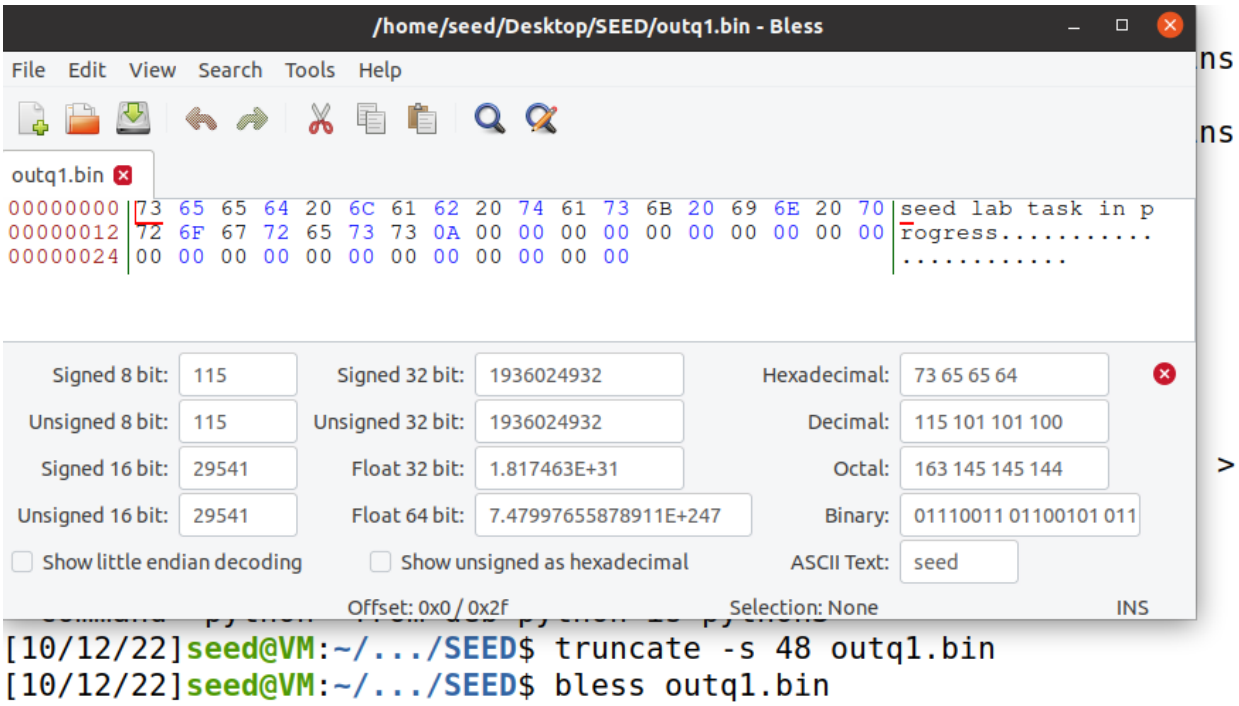



```
ng prefixfile: 'prefix1.txt'
ng initial value: 2dca35cef8ab6762cff635f0fd26a40d

erating first block: .......
erating second block: S01......
ning time: 11.0136 s
/12/22]seed@VM:~/.../SEED$ bless outq1.bin
```

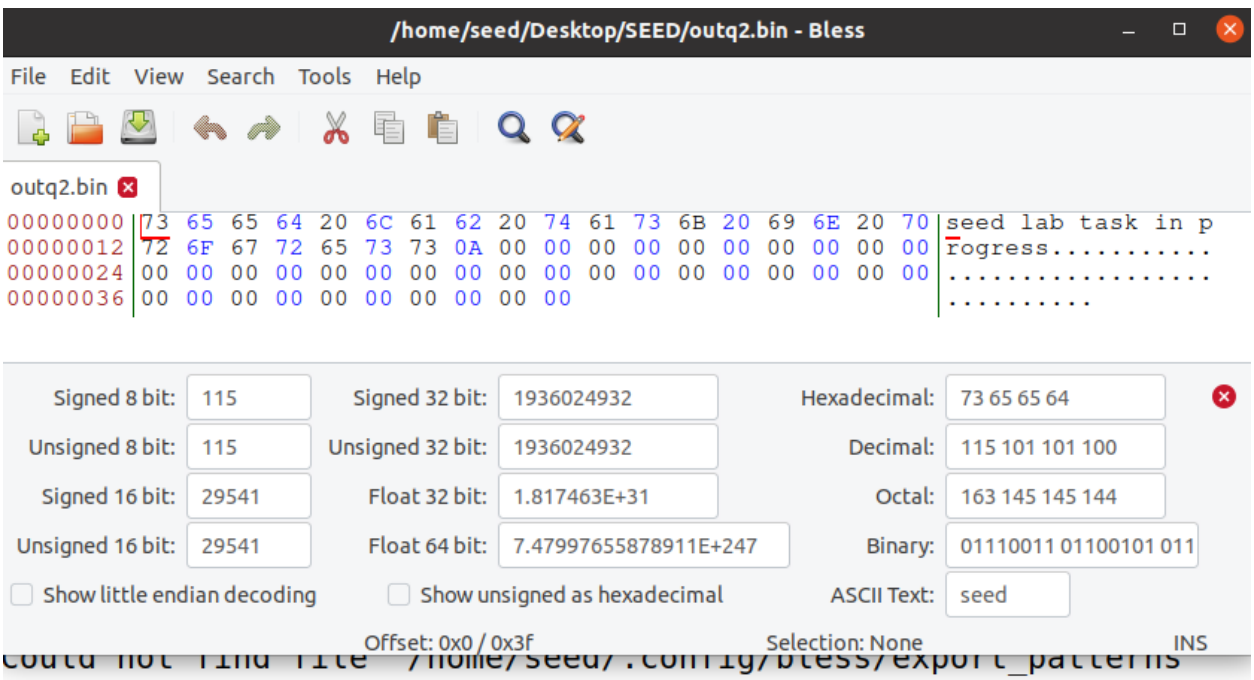



```
ld not find a part of the path '/home/seed/.config/bless/plugins

ld not find a part of the path '/home/seed/.config/bless/plugins

ld not find file "/home/seed/.config/bless/export_patterns"
/12/22]seed@VM:~/.../SEED$ bless outq2.bin
```

- Below are the outputs which have been truncated to 48 bits and 64 bits of the files named *outq1.bin* and *outq2.bin*, respectively.




```
[10/12/22]seed@VM:~/.../SEED$ truncate -s 48 outq1.bin
[10/12/22]seed@VM:~/.../SEED$ bless outq1.bin
```

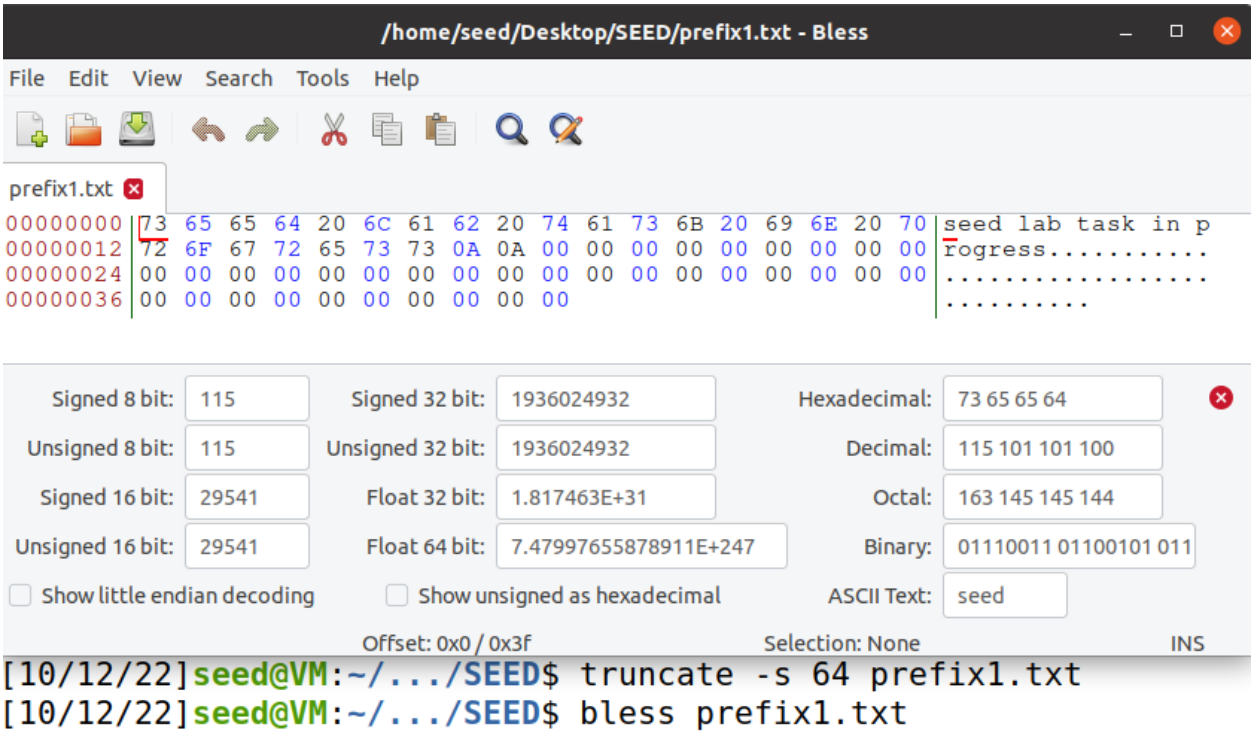



```
[10/12/22]seed@VM:~/.../SEED$ truncate -s 64 outq2.bin
[10/12/22]seed@VM:~/.../SEED$ bless outq2.bin
```

As visible from the differences and truncating the files above, we can notice that to the length will be padded to the provided bits used to truncate.

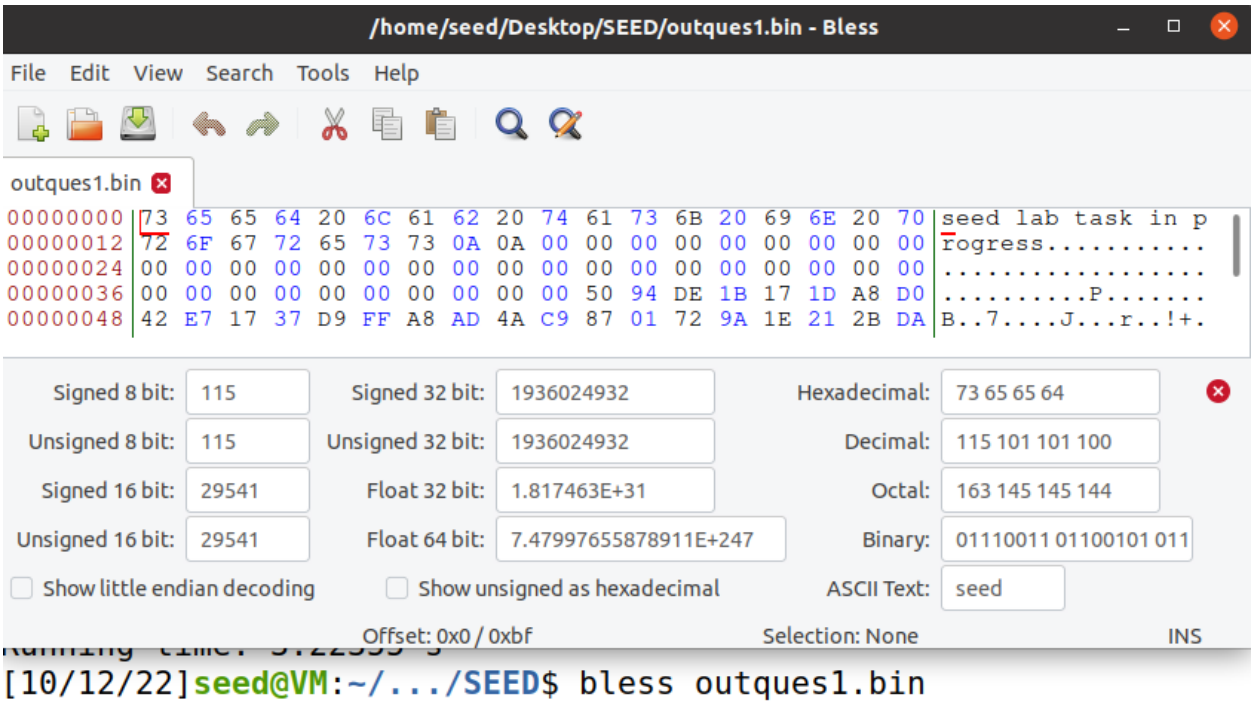
Question 2

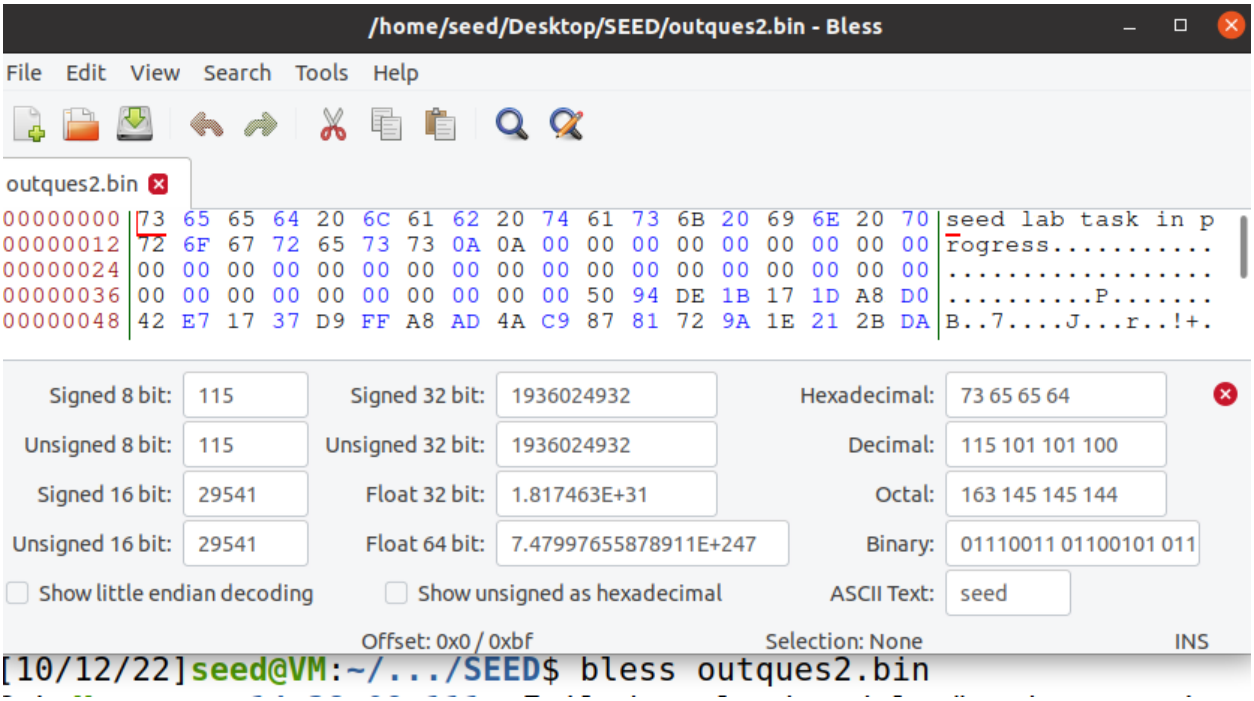
- Here, I have truncated *prefix1.txt* with 64 bits as size value.




```
[10/12/22]seed@VM:~/.../SEED$ truncate -s 64 prefix1.txt
[10/12/22]seed@VM:~/.../SEED$ bless prefix1.txt
```

- I used md5collgen to generate two output files named *outques1.bin* and *outques2.bin*. In order to observe if zero padding still exists.





As evident from above screenshots that there is no padding effect.

Question 3

Not all bytes are different because the bytes only differ at the certain positions but these positions are not constant. Which is evident from the hex editor bless used on both output files.

```
outques1.bin
5 65 64 20 6C 61 62 20 74 61 73 6B 20 69 6E 20 70 72 |seed lab task in pr
7 72 65 73 73 0A 0A 00 00 00 00 00 00 00 00 00 00 00 |ogress............
0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..................
0 00 00 00 00 00 50 94 DE 1B 17 1D A8 D0 42 E7 17 37 |.......P.......B..7
F A8 AD 4A C9 87 01 72 9A 1E 21 2B DA 38 C5 33 0A 10 |....J...r..!+.8.3..
4 BB A1 E9 BA 35 3F 1F 8B 2E 54 5A DF 69 69 54 8F EB |......5?...TZ.iiT..
A 6B 81 87 2A FB 7B 93 5A 8C 69 03 C5 04 EE B1 F8 DD |B.k..*.{.Z.i.......
1 10 EF 6E 56 D4 F4 7F 57 EB 45 85 57 29 E2 E0 13 8B |.Q..nV...W.E.W)....
F 8C 4B 4B 50 45 C1 5A 6A BD 2D 87 A6 8B 56 2D 95 EF |n_.KKPE.Zj.-...V-..
B 50 FD 2A 7C 84 F7 FA 53 08 66 1F C2 0F F4 82 DC 77 |..P.*|...S.f......w
3                                                    |xc
```

```
outques2.bin
00000000 73 65 65 64 20 6C 61 62 20 74 61 73 6B 20 69 6E 20 70 72 |seed lab task in pr
00000013 6F 67 72 65 73 73 0A 0A 00 00 00 00 00 00 00 00 00 00 00 |ogress............
00000026 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..................
00000039 00 00 00 00 00 00 00 50 94 DE 1B 17 1D A8 D0 42 E7 17 37 |.......P.......B..7
0000004c D9 FF A8 AD 4A C9 87 81 72 9A 1E 21 2B DA 38 C5 33 0A 10 |....J...r..!+.8.3..
0000005f 8D D4 BB A1 E9 BA 35 3F 1F 8B 2E 54 5A DF E9 69 54 8F EB |......5?...TZ..iT..
00000072 42 AA 6B 81 87 2A FB 7B 93 DA 8C 69 03 C5 04 EE B1 F8 DD |B.k..*.{...i.......
00000085 AD 51 10 EF 6E 56 D4 F4 7F 57 EB 45 85 57 A9 E2 E0 13 8B |.Q..nV...W.E.W.....
00000098 6E 5F 8C 4B 4B 50 45 C1 5A 6A BD 2D 87 A6 8B 56 2D 95 EF |n_.KKPE.Zj.-...V-..
000000ab F8 EB D0 FC 2A 7C 84 F7 FA 53 08 66 1F C2 0F F4 02 DC 77 |....*|...S.f......w
000000be 78 63                                                   |xc
```

Task 2

- I created two files named file1.txt and file2.txt and inserted the string “seed lab still in progress” in both of them. Followed by md5sum to compare hash values.

```
[10/12/22]seed@VM:~/.../SEED$ echo "seed lab still in progress" >>
file2.txt
[10/12/22]seed@VM:~/.../SEED$ echo "seed lab still in progress" >>
file1.txt
[10/12/22]seed@VM:~/.../SEED$ md5sum file1.txt file2.txt
21413807b76640686ab880cbe460717f  file1.txt
21413807b76640686ab880cbe460717f  file2.txt
```

- Now I concatenated file1.txt and file2.txt and stored the output in file3.txt while comparing md5sum of files. Moreover, I created a new fie named file4.txt with “seed lab could work” string stored in it.

```
[10/12/22]seed@VM:~/.../SEED$ cat file1.txt file2.txt > file3.txt
[10/12/22]seed@VM:~/.../SEED$ md5sum file1.txt file2.txt file3.txt
21413807b76640686ab880cbe460717f  file1.txt
21413807b76640686ab880cbe460717f  file2.txt
84ed723172fa8eb0078b3a16d10ea030  file3.txt
[10/12/22]seed@VM:~/.../SEED$ echo "seed lab could work" >> file4.t
xt
```

- Now I concatenated the files as shown below and compared the md5sum to see the md5sum values of files used as input in concatenation process.

```
[10/12/22]seed@VM:~/.../SEED$ cat file1.txt file3.txt >> file1
[10/12/22]seed@VM:~/.../SEED$ cat file2.txt file3.txt >> file2
[10/12/22]seed@VM:~/.../SEED$ cat file1.txt file4.txt >> file3
[10/12/22]seed@VM:~/.../SEED$ cat file2.txt file4.txt >> file4
[10/12/22]seed@VM:~/.../SEED$ md5sum file1.txt file2.txt file3.txt
file4.txt
21413807b76640686ab880cbe460717f  file1.txt
21413807b76640686ab880cbe460717f  file2.txt
84ed723172fa8eb0078b3a16d10ea030  file3.txt
86d7d042f8f1b54af1308fd2f157f406  file4.txt
```

- Finally I checked the md5sum of output files which obviously shows that for different inputs while adding a suffix (which *is file3.txt* for *file1* and *file2*, and *file4.txt* for *file3* and *file4*) will lead to same md5sum as does for *file1* and *file2* with each other, and *file3* and *file4* for each other.

```
[10/12/22]seed@VM:~/.../SEED$ md5sum file1 file2 file3 file4
8df6f7c21419fdcc51f5a6f160dff20f  file1
8df6f7c21419fdcc51f5a6f160dff20f  file2
b537e7cb8c0bad898d521e96068208c8  file3
b537e7cb8c0bad898d521e96068208c8  file4
```

Hence, the property is proven to be right.

Task 3

- I created a *program.c* file where I stored the provided code in the manual to perform the task.

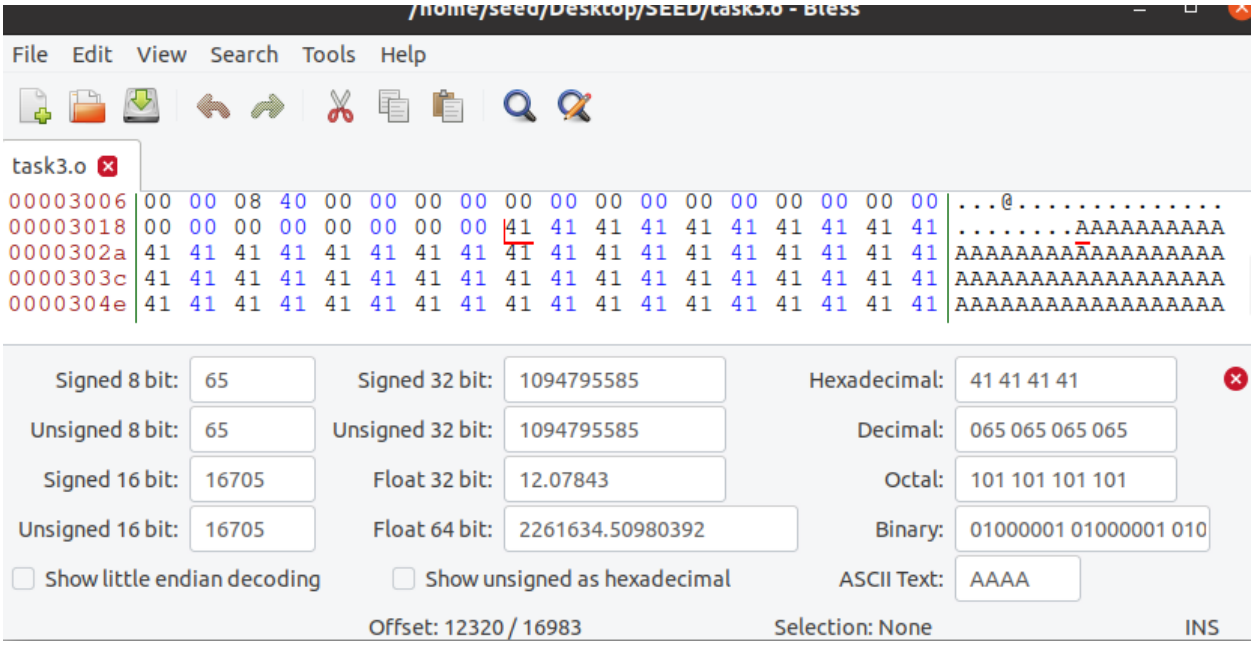
```
[10/13/22]seed@VM:~/.../SEED$ touch task3.c
[10/13/22]seed@VM:~/.../SEED$ gedit
[10/13/22]seed@VM:~/.../SEED$ gedit task3.c
```

- Moreover, I added the Array Values as provided in the manual by using echo and pasting the values in the *task3.c* file.

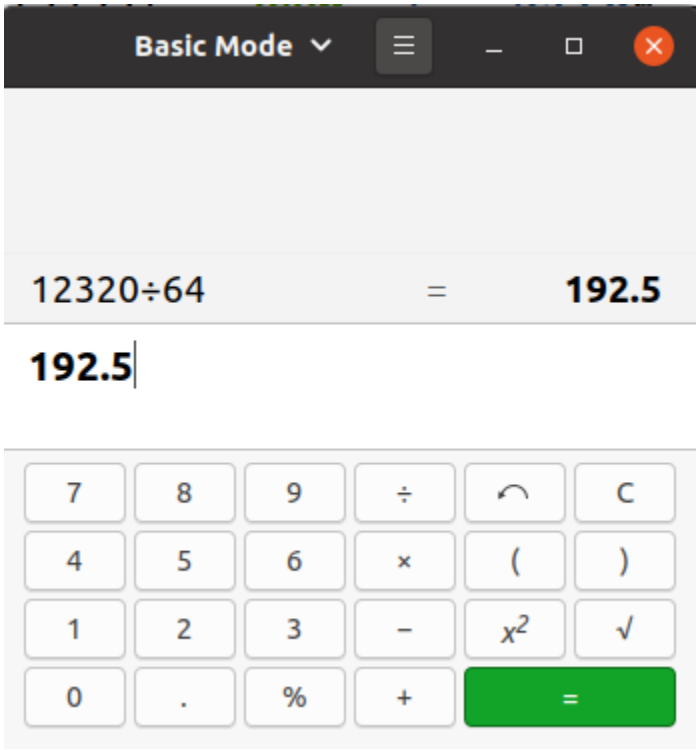
```
[10/13/22]seed@VM:~/.../SEED$ gedit task3.c
^C
[10/13/22]seed@VM:~/.../SEED$ echo "$(python3 -c 'print("0x41,"*199
)')"
0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x
41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41
,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0
x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x4
1,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,
0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x
41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41
,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0
x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x4
1,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,
0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x
41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41
,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0
x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x4
1,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,
[10/13/22]seed@VM:~/.../SEED$ gedit task3.c
```

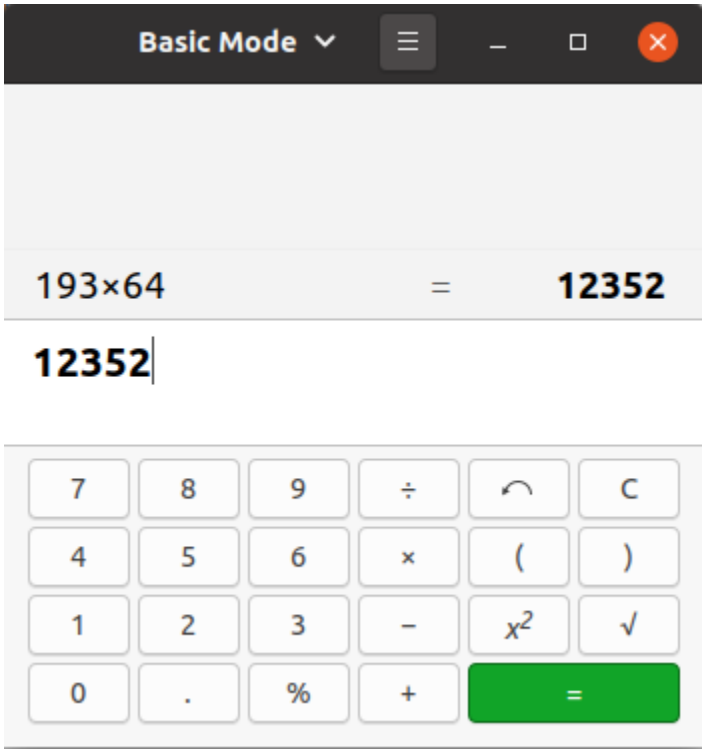
```
Open   task3.c  ~/Desktop/SEED   Save
1 #include <stdio.h>
2 unsigned char xyz[200] = {
3 0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,
4 };
5 int main()
6 {
7 int i;
8 for (i=0; i<200; i++){
9 printf("%x", xyz[i]);
10 }
11 printf("\n");
12 }
```

- Here, I compiled the program and used the output file to see the offset using bless hex editor from where the array values starts. Which seen in decimal is 12320 bytes away from the start of the binary file.



- As per the requirements the length of the prefix should be multiple of 64 so I calculated and found put it was 192. So I created the prefix file named *prefixfile* with byte size 12352 as it is a multiple of 64 about 193 times as we can't use 192.5 or 192 due to the byte reading problems which won't fulfill the requirements.

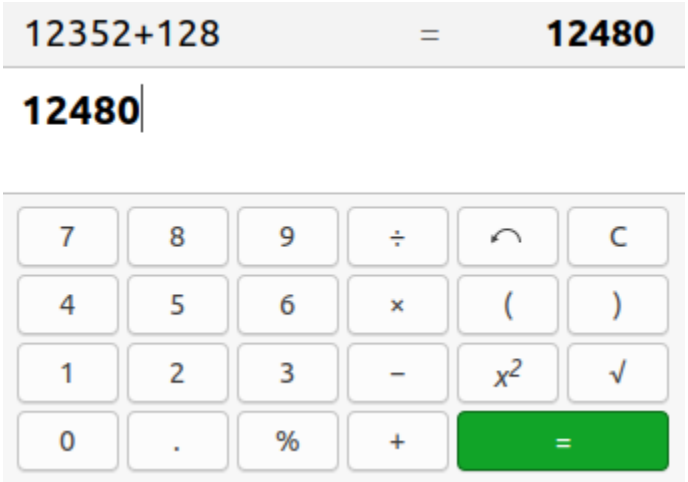




- Then I used the md5collgen to create two output files named *outp1.bin* and *outp2.bin* to compare if the md5sum is the same.

```
[10/13/22]seed@VM:~/.../SEED$ head -c 12352 task3.o >> prefixfile
[10/13/22]seed@VM:~/.../SEED$ md5collgen -p prefix -o outp1.bin out
p2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'outp1.bin' and 'outp2.bin'
Using prefixfile: 'prefix'
Error: cannot open inputfile: 'prefix'
[10/13/22]seed@VM:~/.../SEED$ md5collgen -p prefixfile -o outp1.bin
 outp2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'outp1.bin' and 'outp2.bin'
Using prefixfile: 'prefixfile'
Using initial value: e2c9406dd2f35e08f359d1e5b8842a72

Generating first block: ................................
Generating second block: S00......
Running time: 34.2406 s
```

- Now to add the suffix we will add additional 128 bytes to the prefix size of 12480.



- Here I separated the values of p and q in files named *p* and *q*. Moreover, I created the *sufixfile* containing the suffix according to the above calculations.

```
[10/13/22]seed@VM:~/.../SEED$ tail -c 128 outp1.bin > p
[10/13/22]seed@VM:~/.../SEED$ tail -c 128 outp2.bin > 1
[10/13/22]seed@VM:~/.../SEED$ ls
1     outp1.bin  p           task3.c  Walter
Done  outp2.bin  prefixfile  task3.o
[10/13/22]seed@VM:~/.../SEED$ tail -c 128 outp2.bin > q
[10/13/22]seed@VM:~/.../SEED$ tail -c +12480 task3.0 > sufixfile
```

- After concatenating files as per the formula of “suffix p prefix” in document, I built their executables.

```
[10/13/22]seed@VM:~/.../SEED$ tail -c +12480 task3.o > sufixfile
[10/13/22]seed@VM:~/.../SEED$ cat prefixfile p sufixfile > task31
[10/13/22]seed@VM:~/.../SEED$ chmod +x task31
[10/13/22]seed@VM:~/.../SEED$ cat prefixfile q sufixfile > task32
[10/13/22]seed@VM:~/.../SEED$ chmod +x task32
```

- When executing these files we get the same output

```
[10/13/22]seed@VM:~/.../SEED$ ./task32
4141414141414141414141414141414141414141414141414141414141414141233
0ec2d4e9258e91c874c85b65d6c9093837b2bf1582fc308a40452cbc2d6055cfdf4
ecc2b618b204d63342a1eb4f3b9a2105ad1135d6f58d6a476225eeb60552b696e12
91c39096c4bb8c0642a833a82fc7415dd55714c5d01859a24f919f269b67594b189
aceff8f24bcab4185f33c4b53923321b0e3d3fd237d4141414141414141414141414
1
4141414141414141414141414141414141414141414141414141414141414141
[10/13/22]seed@VM:~/.../SEED$ ./task31
4141414141414141414141414141414141414141414141414141414141414141233
0ec2d4e9258e91c874c85b65d6c9093837babf1582fc308a40452cbc2d6055cfdf4
ecc2b618b204d6334aa1db4f3b9a2105ad1135d6f58d68a476225eeb60552b696e1
291c39096c4bb8c0642a833a2fc7415dd55714c5d01859a24f919f269b67594b189
aceff8724ccab4185f33c4b53923321b063d3fd237d4141414141414141414141414
1
4141414141414141414141414141414141414141414141414141414141414141
```

- Finally I moved to check md5sum which also was the same. Hence, proving the point of two files generating same md5sum.

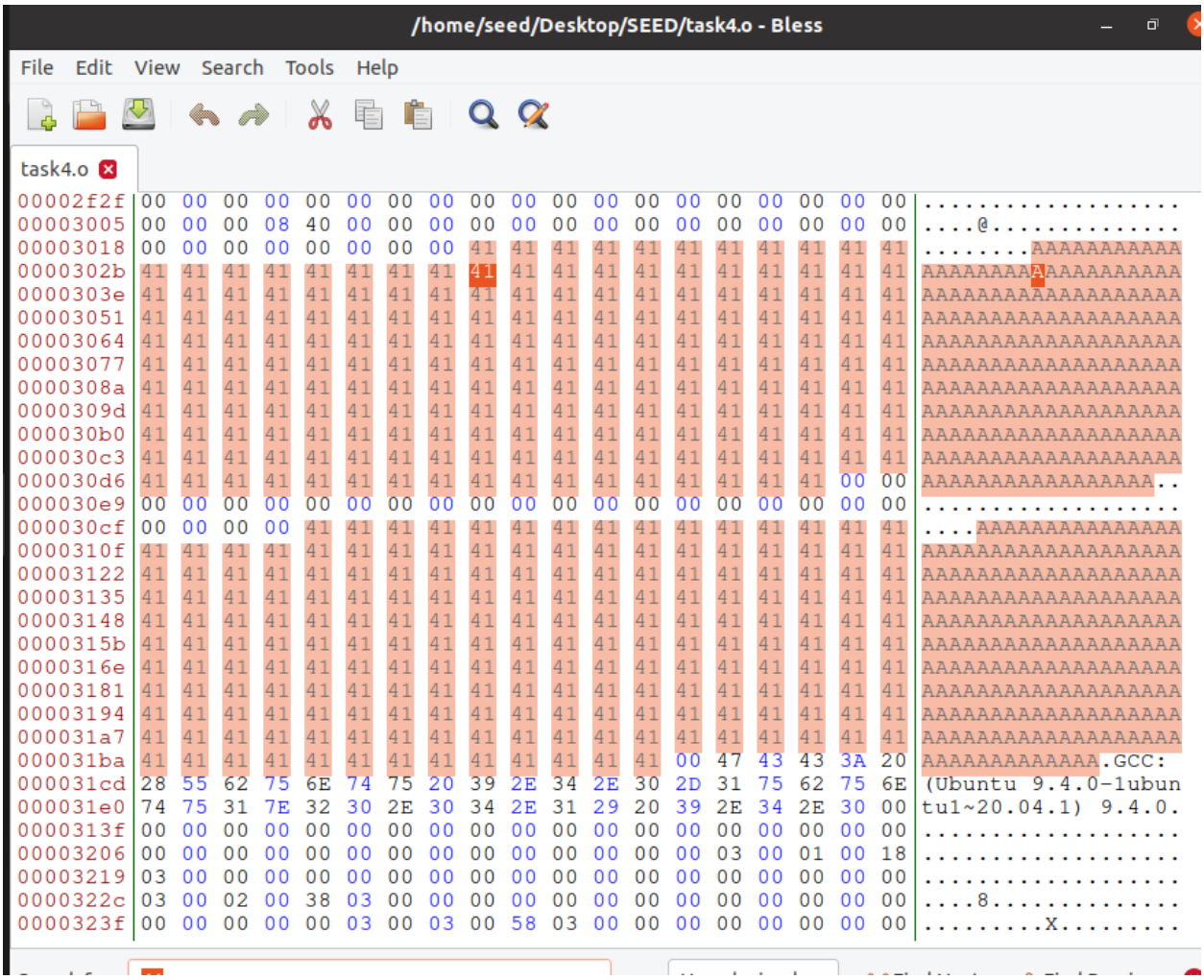
```
[10/13/22]seed@VM:~/.../SEED$ md5sum task31 task32
863c1f5223fcdcf0d3e26a03da1b818f  task31
863c1f5223fcdcf0d3e26a03da1b818f  task32
```

Task 4

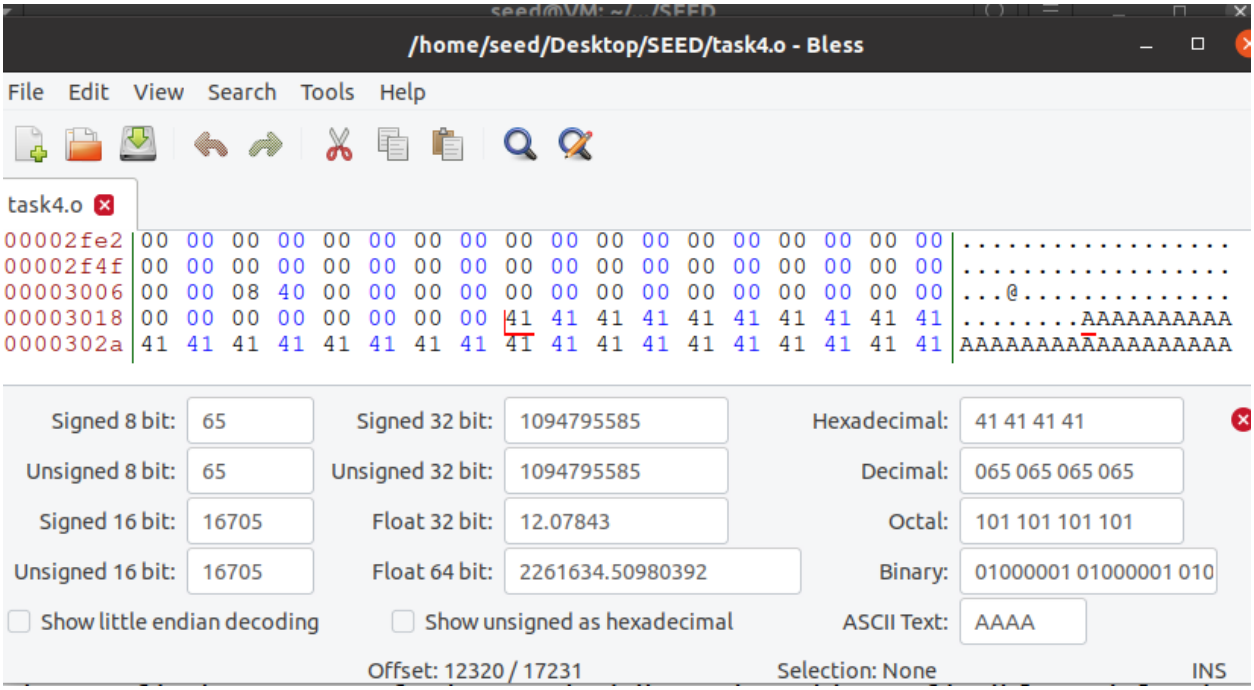
- I made a file named *task4.c* and stored the provided code with modifications left to us.

```c
#include <stdio.h>

 unsigned char x[200] = {

0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0

};

unsigned char y[200] = {

0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0x41,0

 };

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

 if(i == 200){ printf("%s", "benign code"); } /* x = y */
 else{ printf("%s", "WARNING: malicious code"); } /* x != y */

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

- After compiling *task4.c* in output file task4.o, I checked the hex in bless hex editor of *task4.o*. And we can observe two arrays in the screenshot below.



- Given that the Array starts at offset 12320 we will use it here as it is.



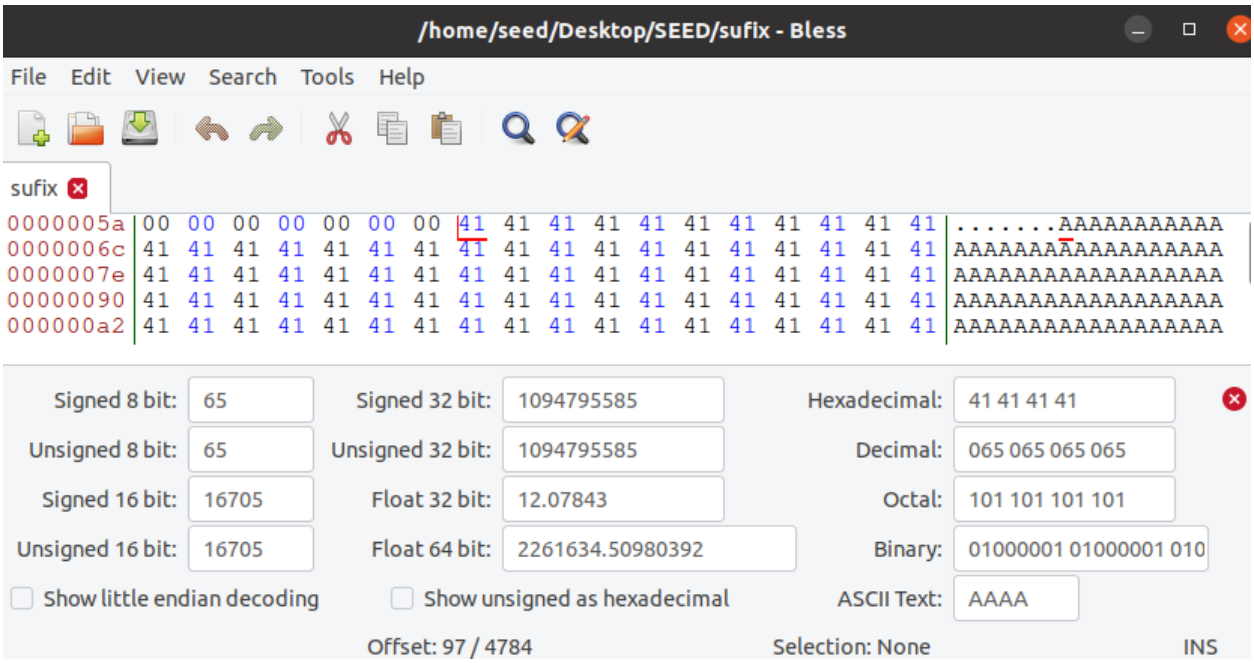
- Now creating the prefix and using md5collgen to create two output files.

```
[10/13/22]seed@VM:~/.../SEED$ head -c 12320 task4.o > prefix
[10/13/22]seed@VM:~/.../SEED$ 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: 67a28b17bb7a43620c53f2012f1dbb1d

Generating first block: ..........................................
.
Generating second block: S00...
Running time: 56.4522 s
```

- Creating p,q and suffix files.

```
[10/13/22]seed@VM:~/.../SEED$ tail -c 128 out1.bin > p
[10/13/22]seed@VM:~/.../SEED$ tail -c 128 out2.bin > q
[10/13/22]seed@VM:~/.../SEED$ tail -c +12448 task4.o > sufix
```

- Now checking the hex value of *sufix* file which is 97 in decimals.



- Taking offset value 97 bytes. Then I performed head and trail operations, and concatenated in order to perform the operation as provided in the document. By doing this the file named *task41* is declared benign and the *task42* is declared malicious.

```
[10/13/22]seed@VM:~/.../SEED$ head -c 96 sufix > sufix1
[10/13/22]seed@VM:~/.../SEED$ tail -c +225 sufix > sufix2
[10/13/22]seed@VM:~/.../SEED$ cat prefix p sufix1 p sufix2 > task41
[10/13/22]seed@VM:~/.../SEED$ cat prefix q sufix1 p sufix2 > task42
[10/13/22]seed@VM:~/.../SEED$ chmod +x task41
[10/13/22]seed@VM:~/.../SEED$ ./task41
benign code
[10/13/22]seed@VM:~/.../SEED$ chmod +x task42
[10/13/22]seed@VM:~/.../SEED$ ./task42
WARNING: malicious code
```

- We achieved the results as the md5sum is the same as desired even though difficult to achieve.

```
[10/13/22]seed@VM:~/.../SEED$ md5sum task41 task42
c9ab5ef39b67e6ac772a8f8137cb839d  task41
c9ab5ef39b67e6ac772a8f8137cb839d  task42
```