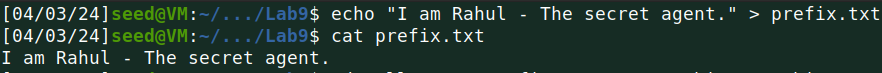
**Lab 9: MD5 Collision Attack Lab**

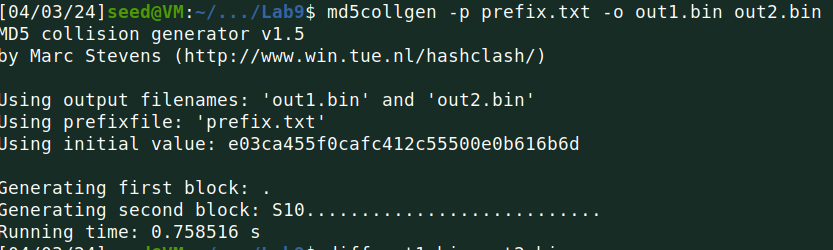
**Task 1:**

In this task, I have to generate two different files with the same MD5 hash values.

First I created a prefix.txt file and added some text to that.

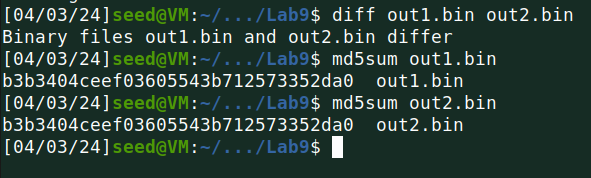


Now, I ran the following commands to generate two output files out1.bin and out2.bin.



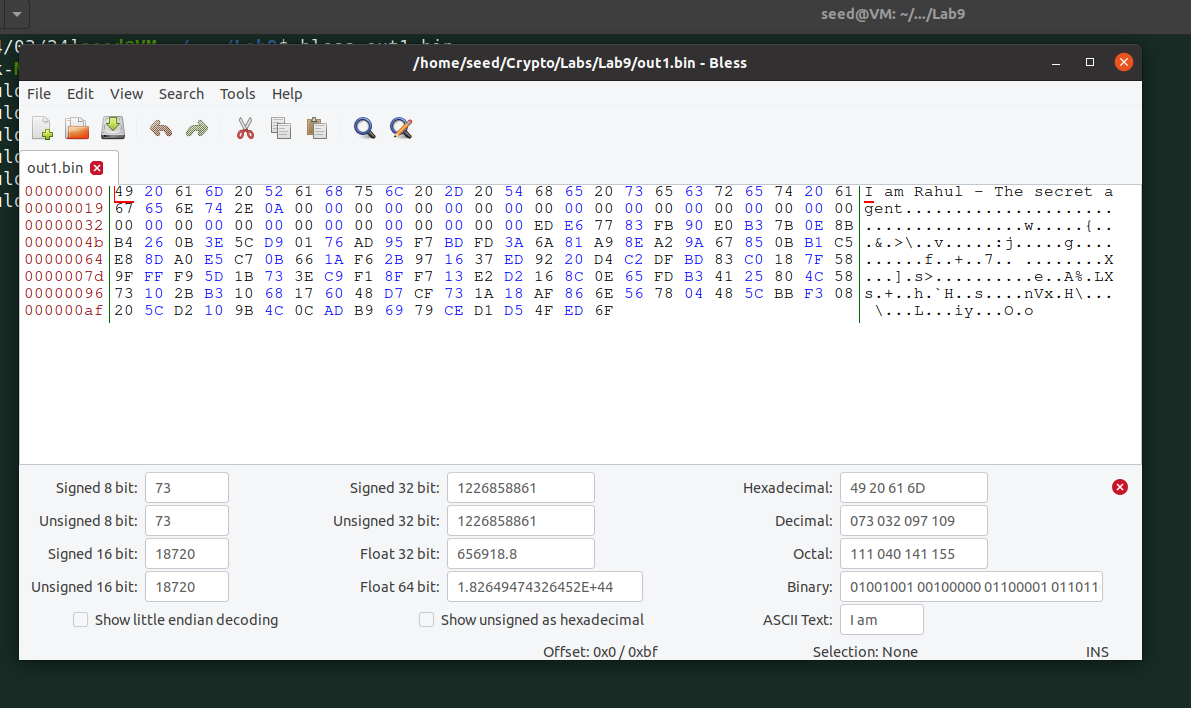
After that, I used the diff command for both files to check if they were different or not.

And as two files were different immediately I used the md5sum command to check the md5 hash of each file.

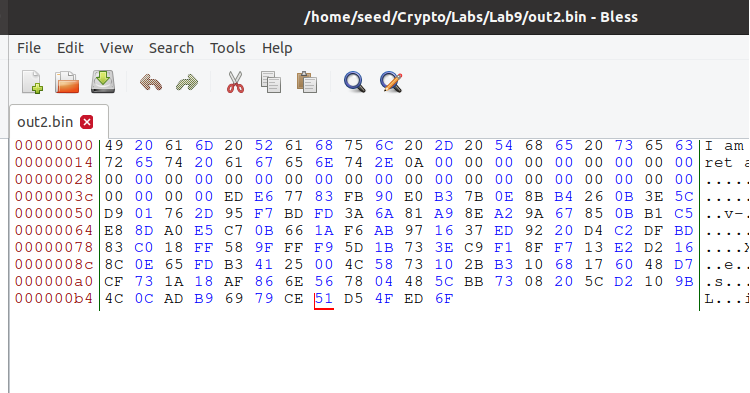


Below is the visual presentation of out1.bin file using bless command.

**out1.bin**



**out2.bin**



As you can see, in the out2.bin file marked location is different from the out1.bin file location.

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

Answer:

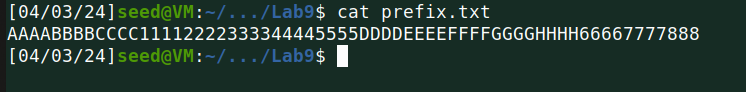
If the prefix file's length is not a multiple of 64, md5collgen will add padding to extend it to a multiple of 64.

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

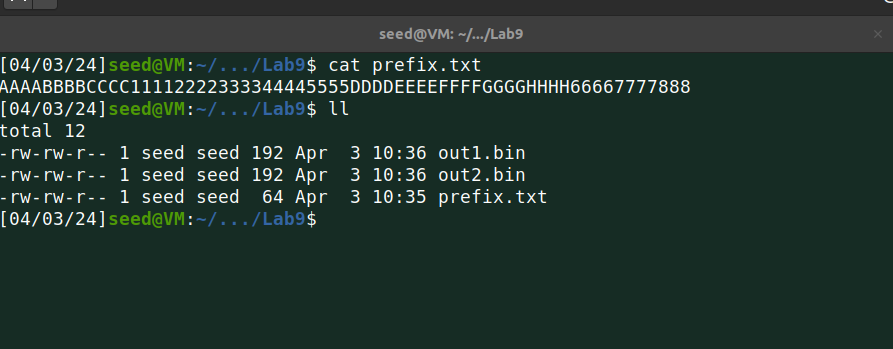
Answer:

For this question, I need to change the prefix.txt file to match the size of 64 bytes.

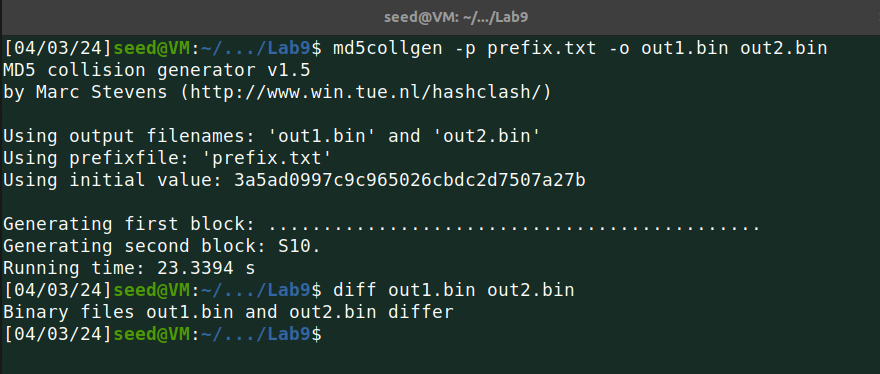
Below is the content of a prefix.txt file.



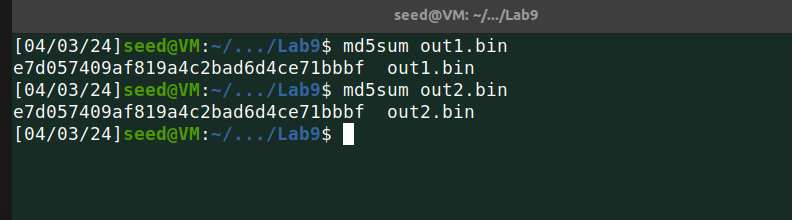
Here is the screenshot showing that the length is 64 bytes.



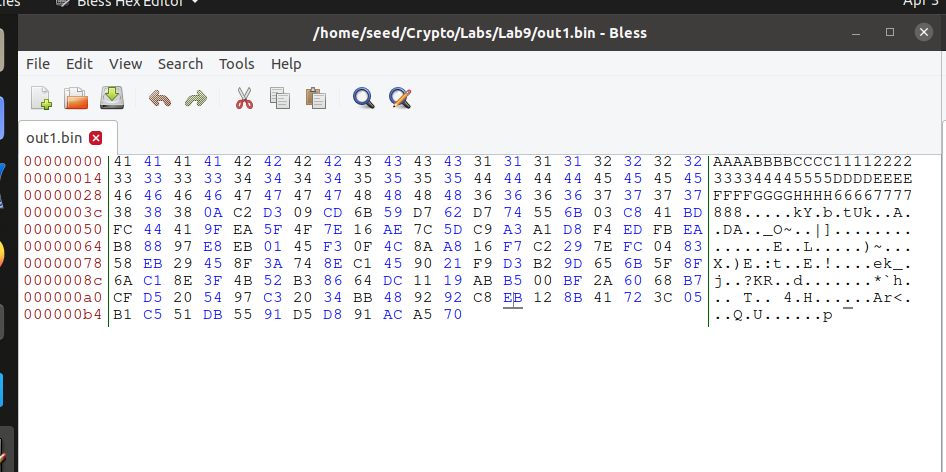
Now I ran the same commands as previously used to generate two output files and check whether both are different or not!



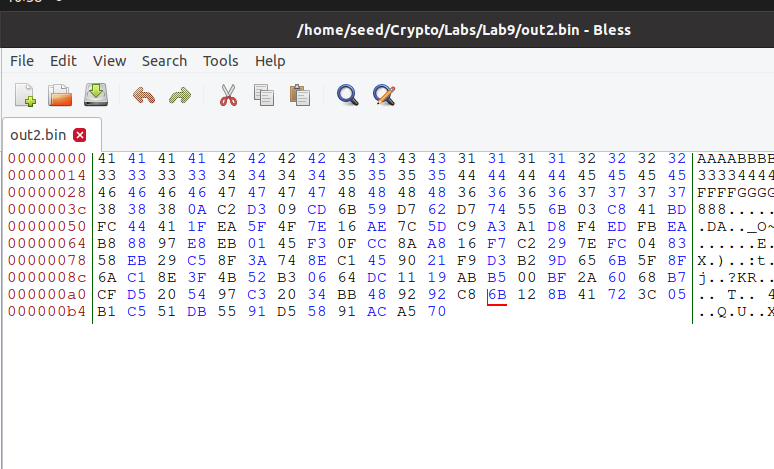
As the files were different, now I generated hash values for both files using the below.



**out1.bin**



**out2.bin**

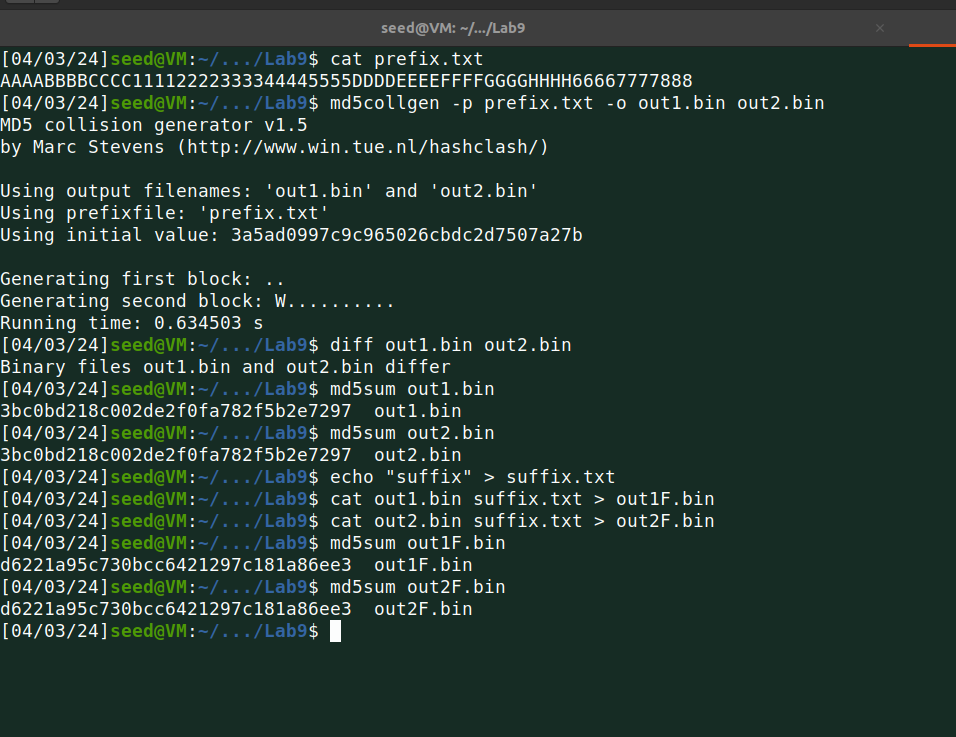


As the screenshot shows the bless output of both output files, the marked location is different.

**Task 2:**

In this task, I generate two output files from a prefix and then append a common suffix to both. This demonstrates that adding the same suffix, T, to two input files with identical hash values will result in two outputs that also have the same hash value.

I used the same steps as used in task 1 to generate two output files.

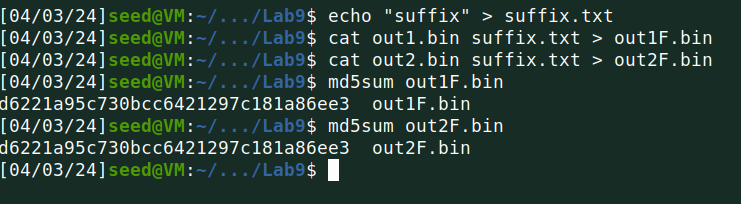


Now, I used the echo command to generate a suffix file.

Then using the cat command concatenated the out1.bin and suffix.txt files into a new out1F file, and did the same for the out2.bin file.

And last generated hash value for new output files.

All commands are one by one below.



This confirms that they are indeed identical. Thus, the property has been successfully demonstrated by me.

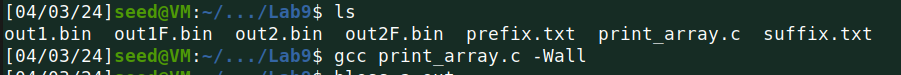
**Task 3:**

In this task, I am provided with a C program. My objective is to develop two distinct versions of this program, ensuring the contents of their xyz arrays differ, yet the hash values of the executables remain identical.

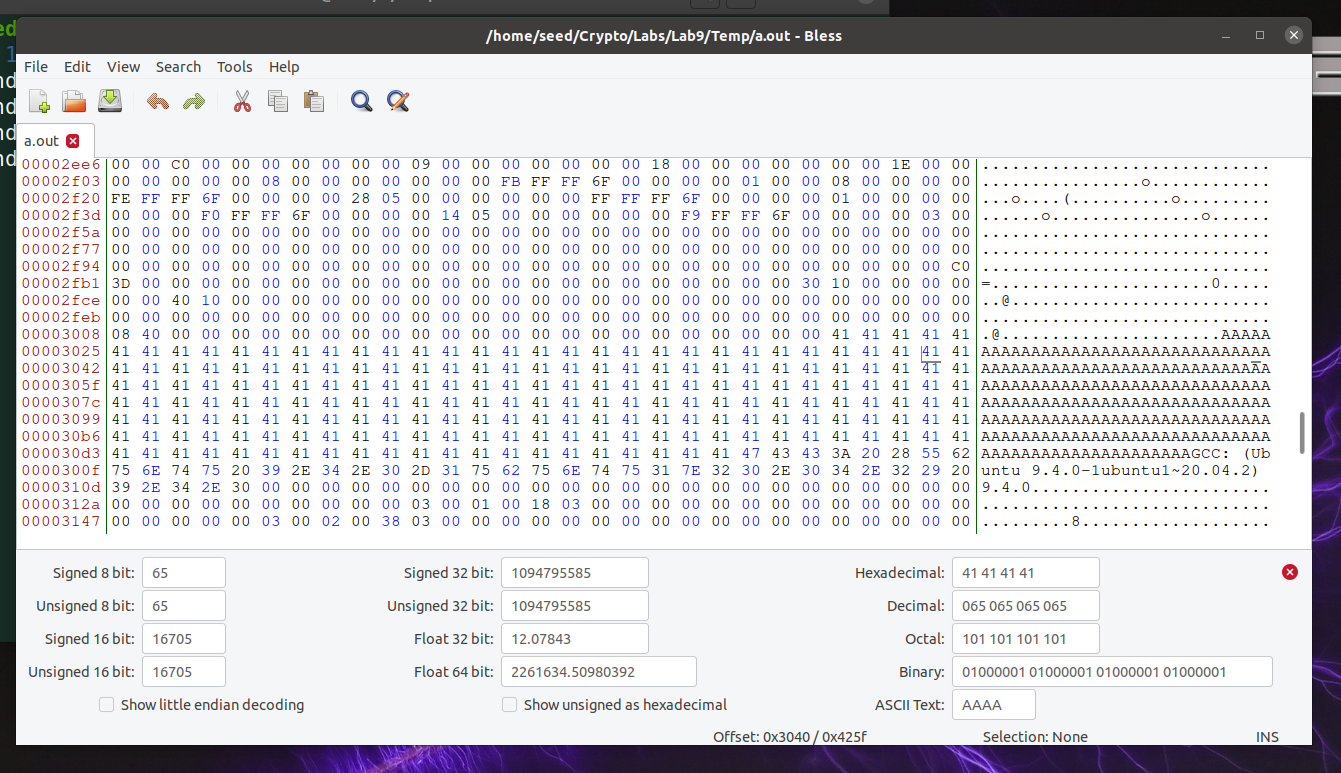
**Code:**

| #include <stdio.h>  unsigned char xyz[200] = {  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,  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,  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,  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,  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,  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,  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41 };  int main() {  int i;  for (i=0; i<200; i++){  printf("%x", xyz[i]);  }  printf("\n"); } |
| --- |

I ran the code using below.



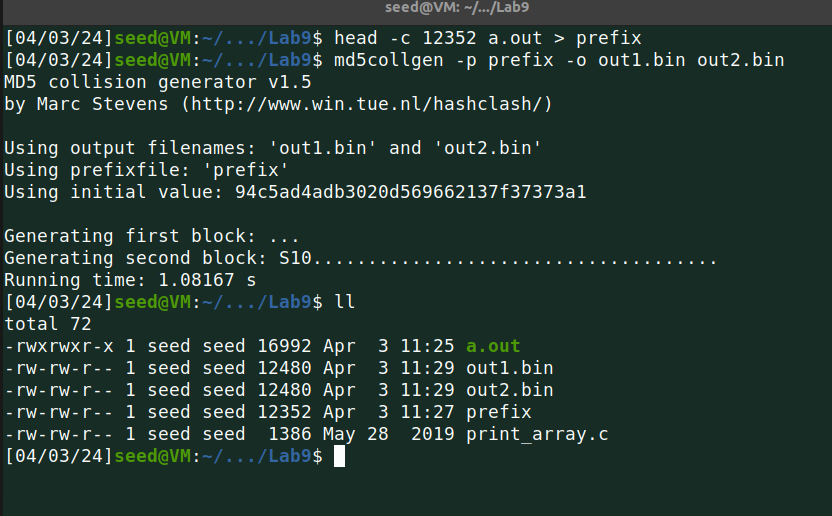
Then open the a.out file using bless….



Using bless output I found the required offset value to be 0x3040.

Now I used the head command using the offset value to generate the prefix.

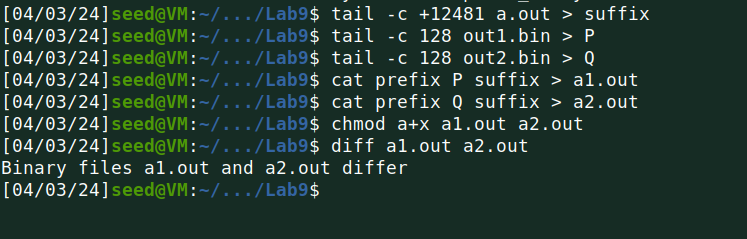
And then using md5collgen to generate two output files with the same hash value.



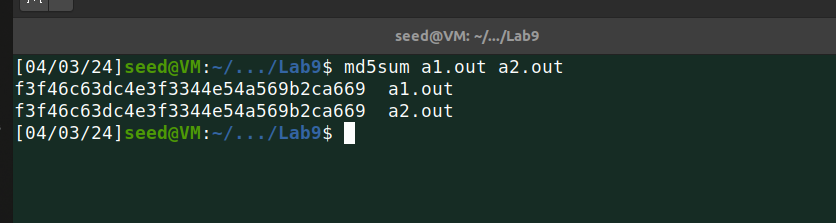
As the screenshot shows, the size of the files. Binary files have the size of 12480 bytes and the prefix is with 12352 bytes.

So I started from 12481 for both output files.

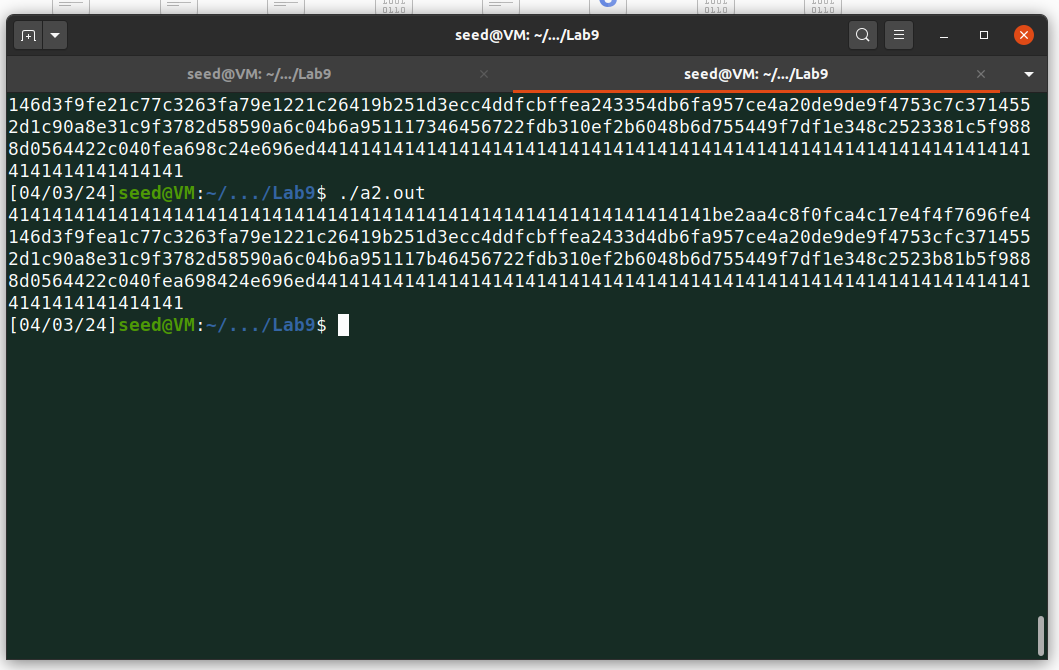
1. Generated suffix using the offset value.
2. Generated P & Q
3. Concatenated prefix, P, and suffix to a1.out
4. Concatenated prefix, Q, and suffix to a2.out
5. Gave the permission to both new executables files.
6. Checked whether they are different or not.



Using md5sum to calculate the md5 hash value from both output files.



Below is the output of both files.



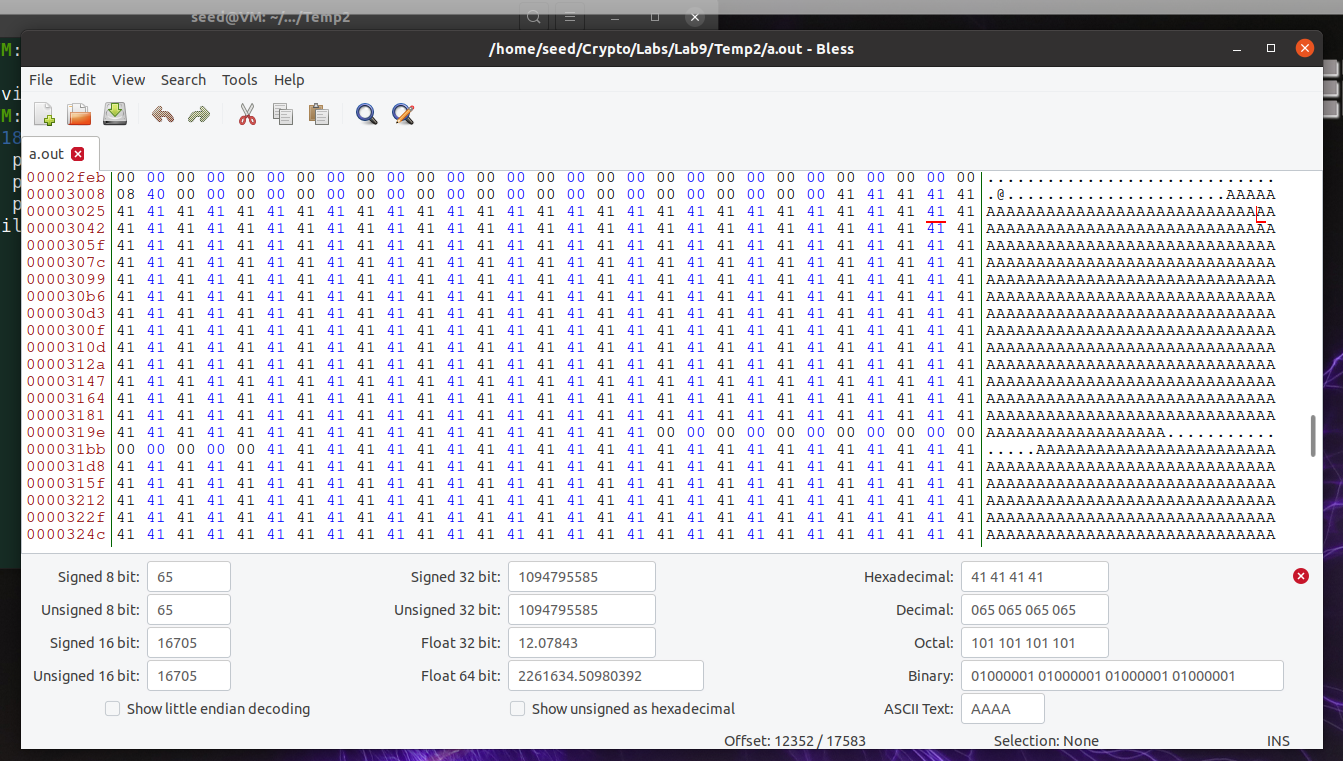
**Task 4:**

In this task, employing the program provided, I create two versions of the same program, both with identical hash values. I develop two programs that have the same MD5 hash; however, one executes benign instructions, while the other carries out malicious instructions.

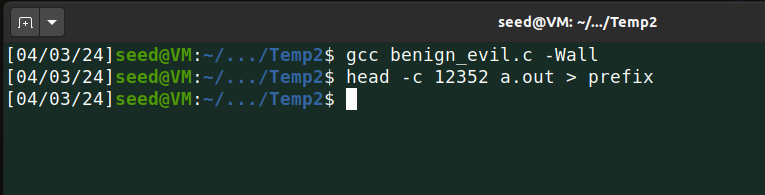
**Code:**

| #include <stdio.h> #define LENGTH 400  unsigned char X[LENGTH]= {  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" };  unsigned char Y[LENGTH]= {  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" };  int main() {  int i = 0;   for (i =0; i< LENGTH; i++){  if (X[i] != Y[i]) break;  }  if (i==LENGTH){   printf("%s\n", "Executing benign code... ");  }  else {   printf("%s\n", "Executing malicious code... ");  }  return 0; } |
| --- |

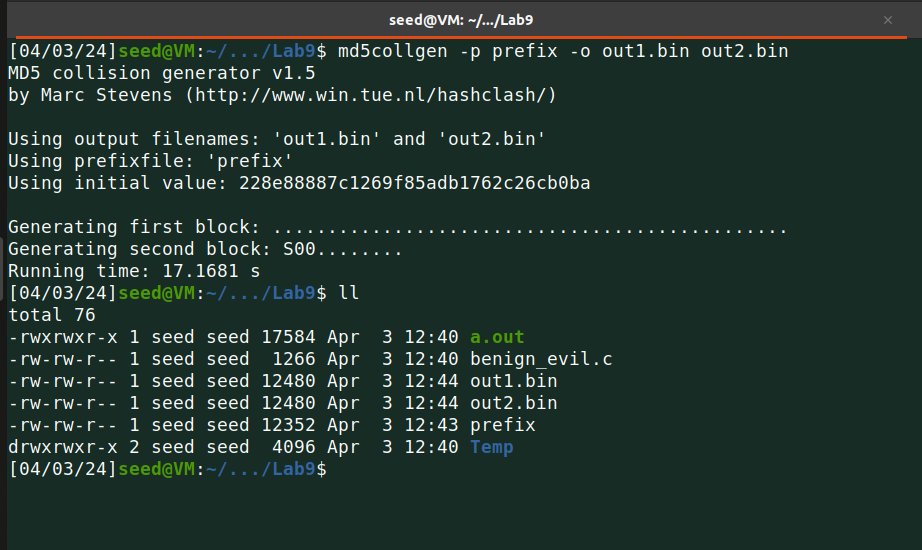
I ran the code and then opened the a.out file using bless.



Then, I select a position that is a multiple of 64. As shown in the screenshot, I have chosen 12352. I will use the first 12352 bytes of a.out to copy into the prefix file.



Then as per steps I used md5collgen to generate two output files and checked the size of the files.



And now execute the steps as below one by one:

1. The binary files have a size of 12480 bytes and the prefix has a size of 12352 bytes, so I started the suffix from 12481.
2. I need to overwrite the subsequent 128 bytes of the array with 'P' in the first file and with 'Q' in the second file. Therefore, I use the tail command to achieve this (12352 + 128 = 12480).
3. The suffix array contains suffix\_1, P/Q, and suffix\_2. Thus, I overwrite the initial 288 bytes with suffix\_1 and everything except the first 417 bytes with suffix\_2.
4. The further step involves merging the files we've prepared into two output files. The first file combines the prefix, P, suffix\_1, another P, and suffix\_2. The second file consists of the prefix, P, suffix\_1, Q, and suffix\_2.
5. Gave the permission to execute both files.
6. And finally, I can see the output of both files.

