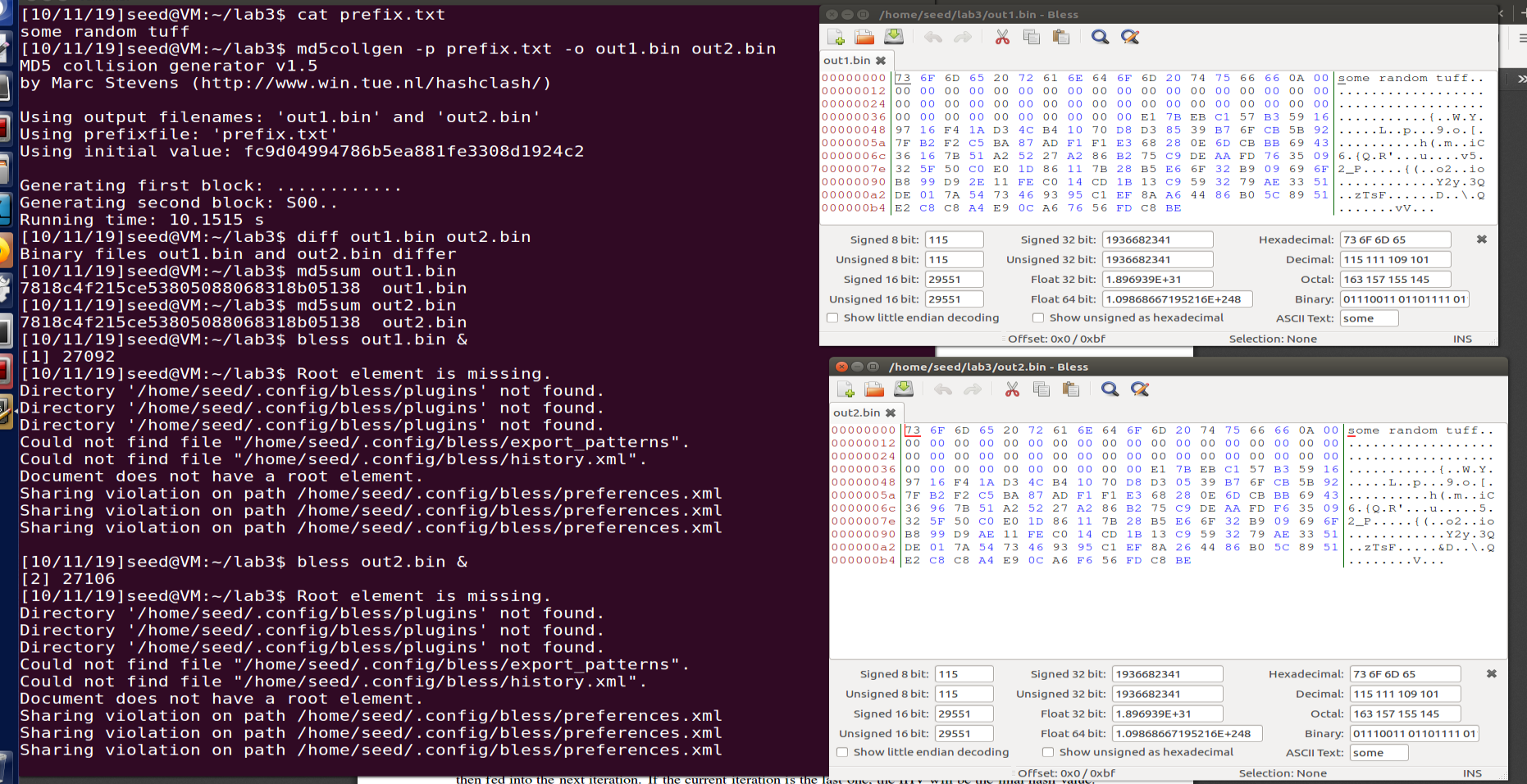
**LAB3: Hash Pramod kumar, pjk5502@psu.edu**

**Task 1:**

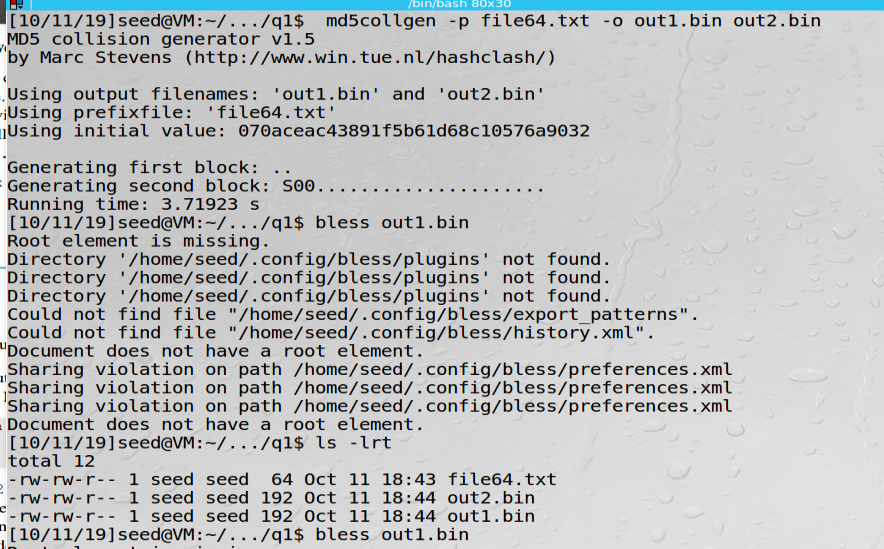
Execution :

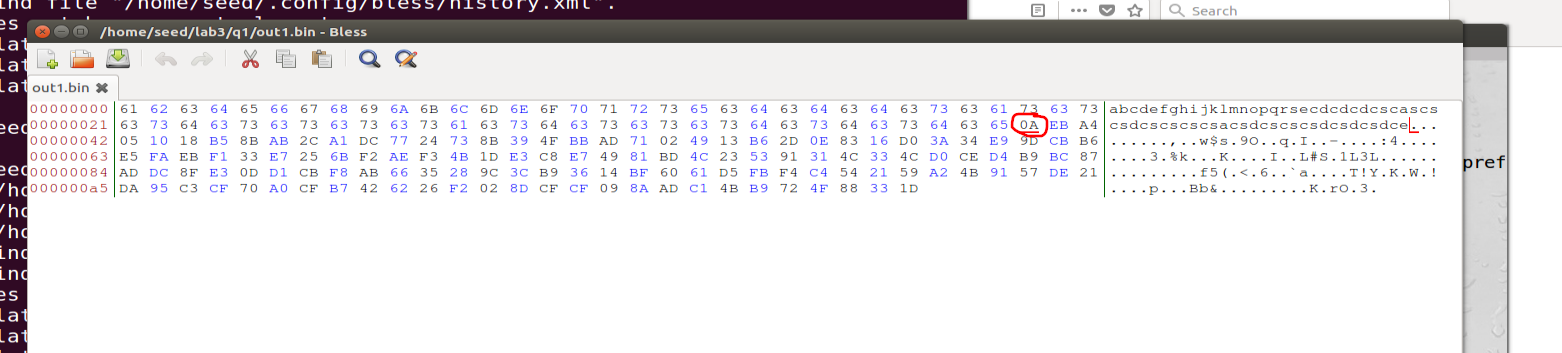


Observation

**Subtask 1**: when file size is less then 64 or (448 bit mod 512bit), rest of byte in are padded with one OA and followed by (OO)\* .

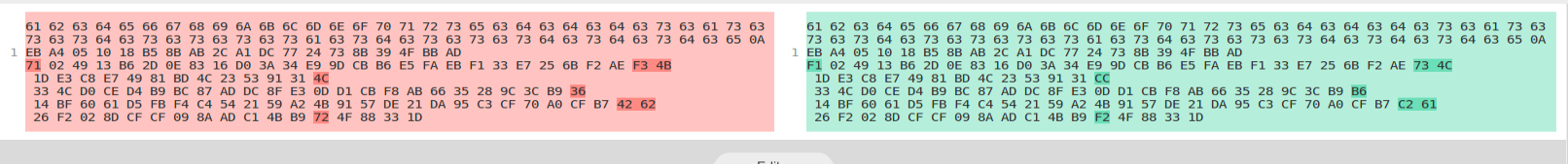
**Subtask 2**: Created a file with 64 byte, As we can see it doesn’t have long padding but it does have 1 of padding.





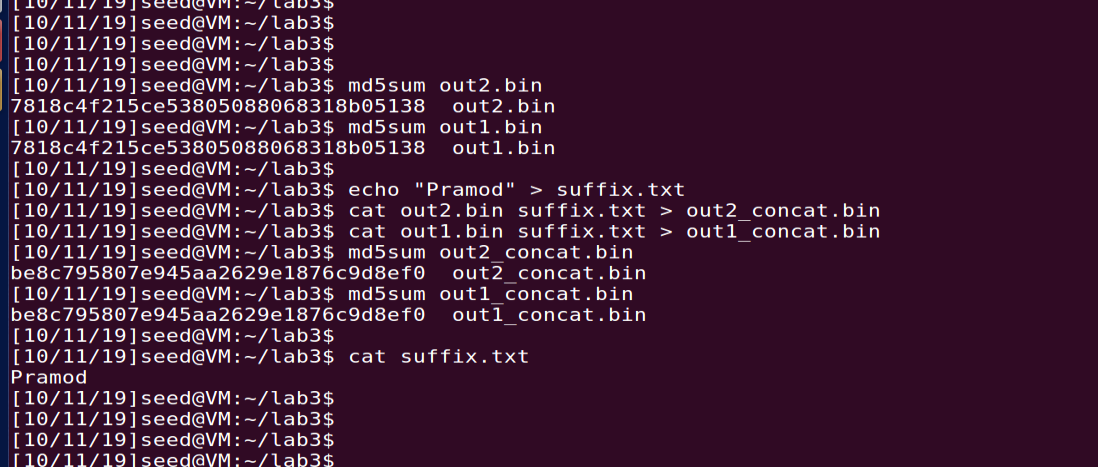
**Subtask 3 :**  those file has same hash but there is 8 bytes are different.

71,F3,4B,4C,36, 42, 62, 72

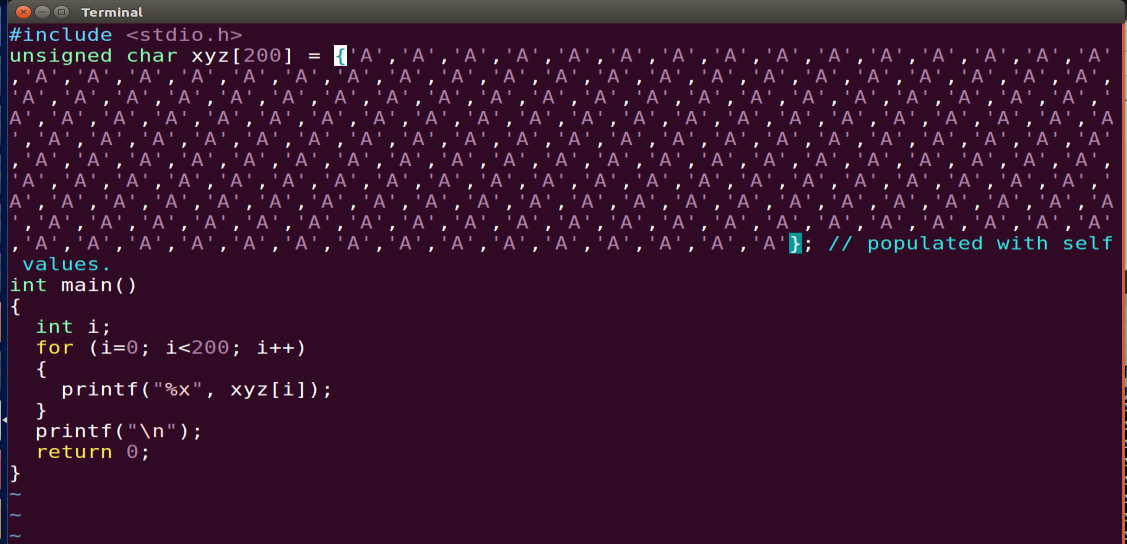


**Task 2:**

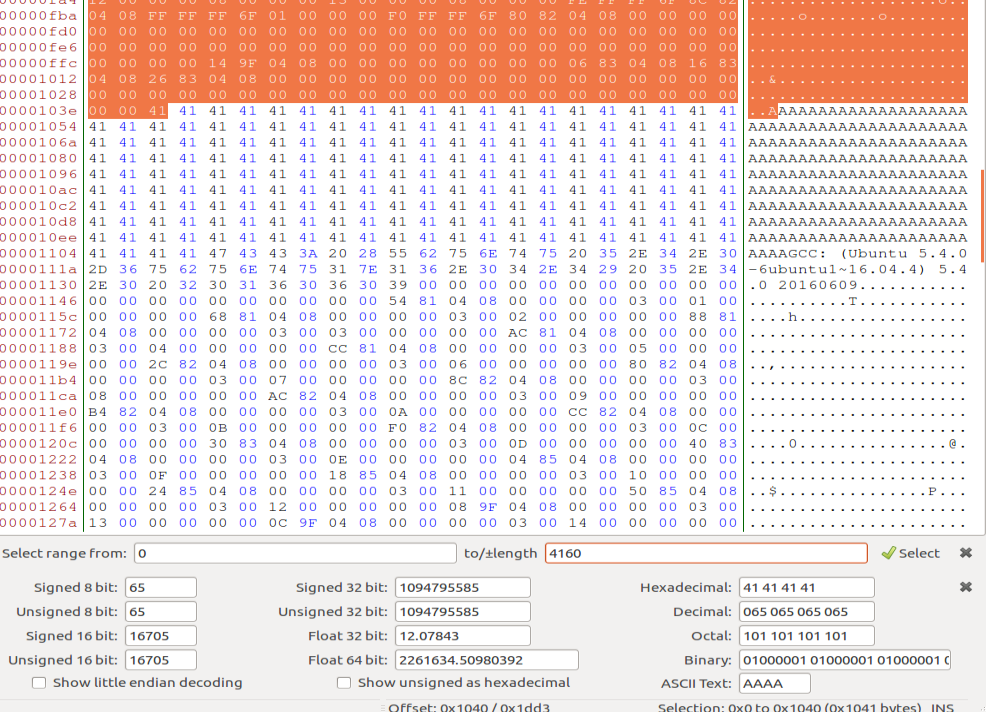
Create a string file suffix.txt. concatenate with out1.bin and out2.bin file from task1 as they have same hash function. On checking md5 of new files it proves the property



Task 3: Write a program which print 200 “A” and compile with gcc to generate executable.

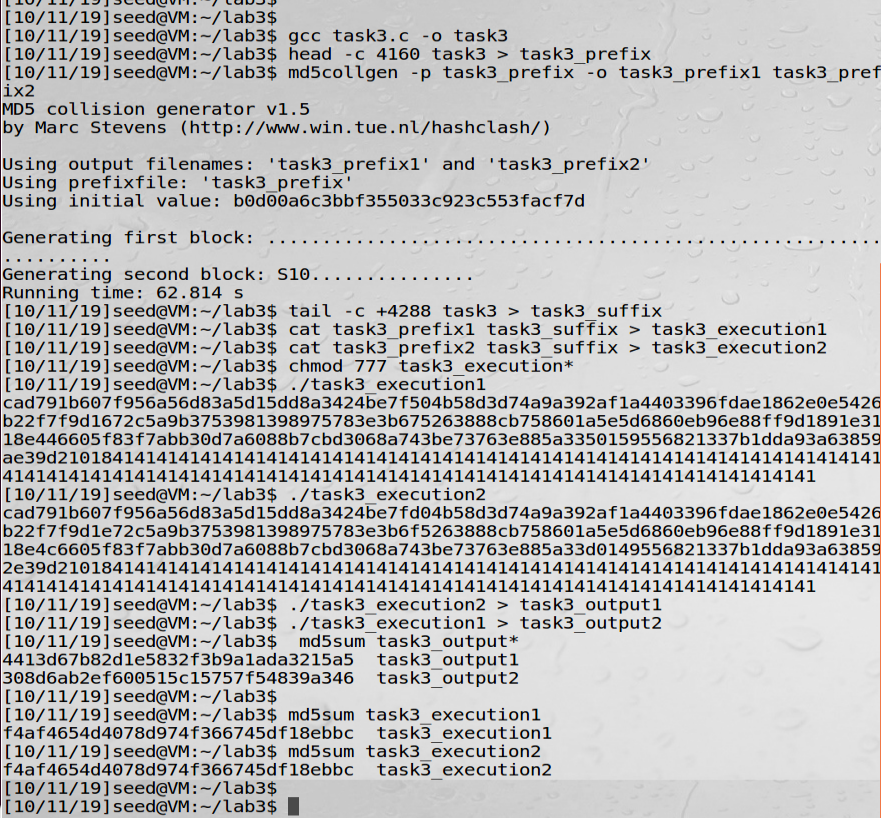


Open executable, As we can see in bless, 1st a is 0x1040 offset(which is multiple of 64). So we can use this as PREFIX.



Creating prefix and its md5collsion blocks

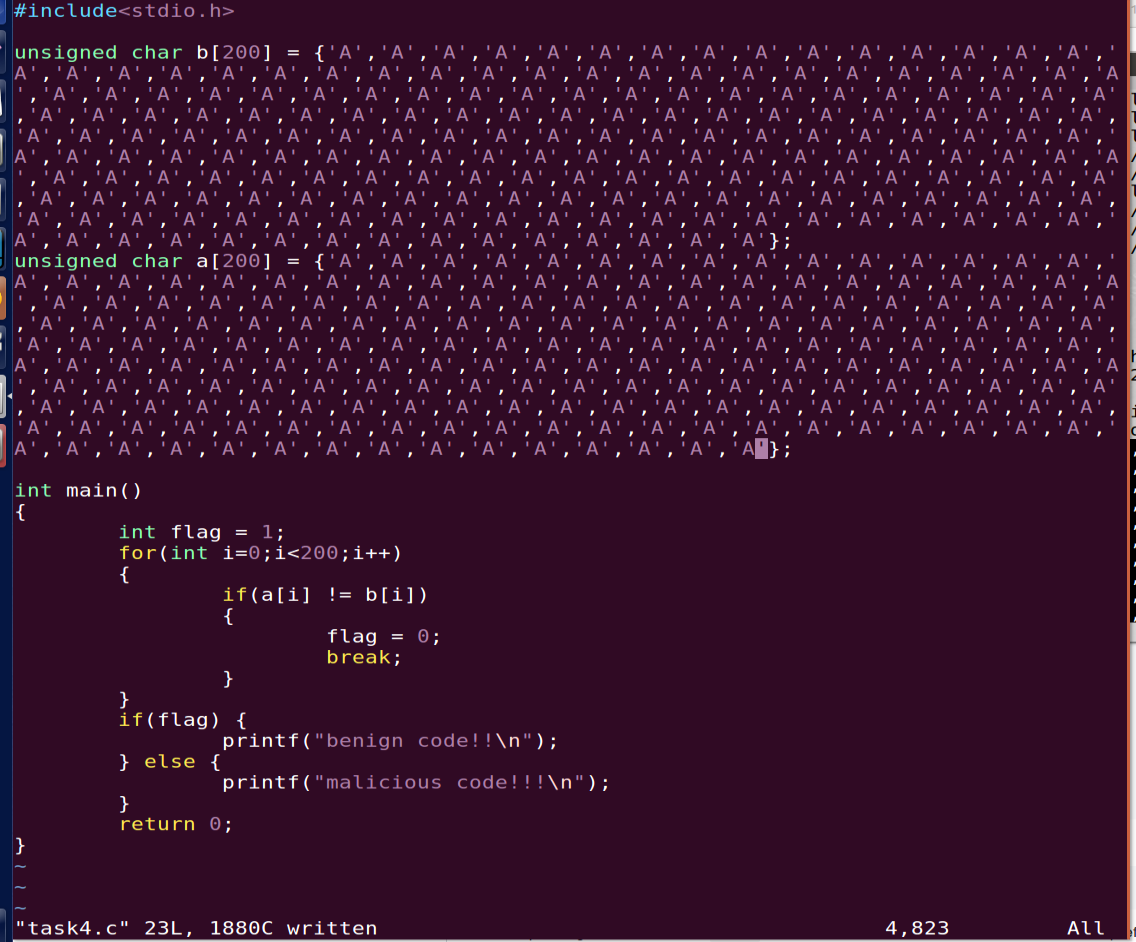
Now lets create suffix after leaving 128 and create 2 New executable and run the program



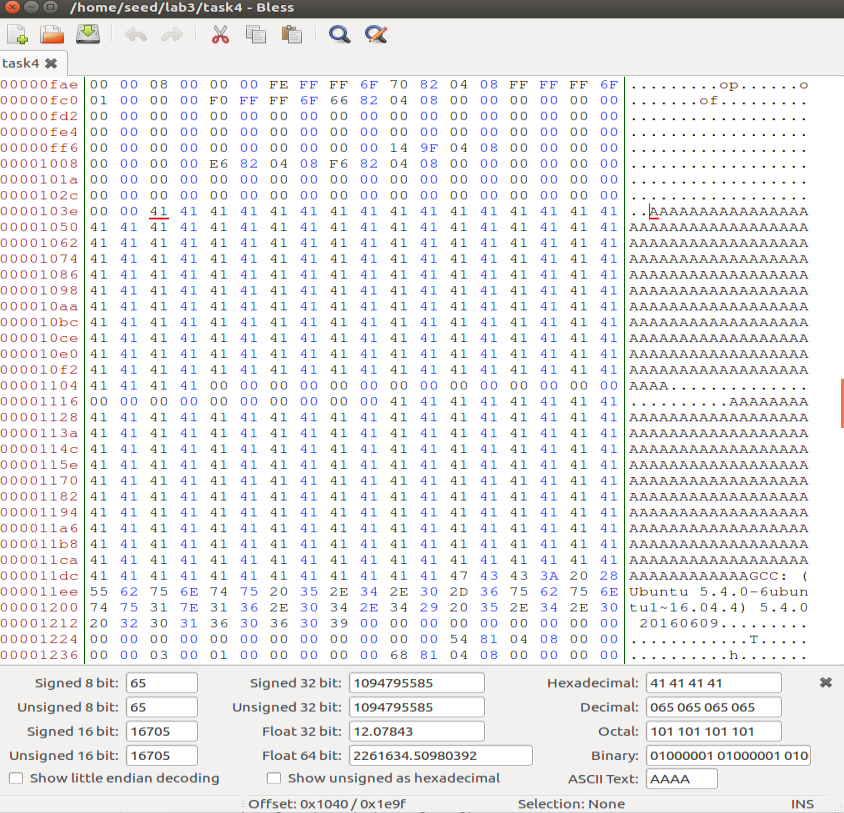
As we can see that “task3\_execution1 and task3\_execution2” has same md5 but output of both is different and we can validate with md5 of output of both executions.

**Task4:**

Create a program which has 2 array and check those array bitwise. If both are same then execute one path else other path.. according to below code, both array contain same elements



Bless output to show the output of executable



Create prefix, md5 hash collision block and suffix. But this time lets take 66\*64 (4224) as offset



We have created Suffix in previous step.

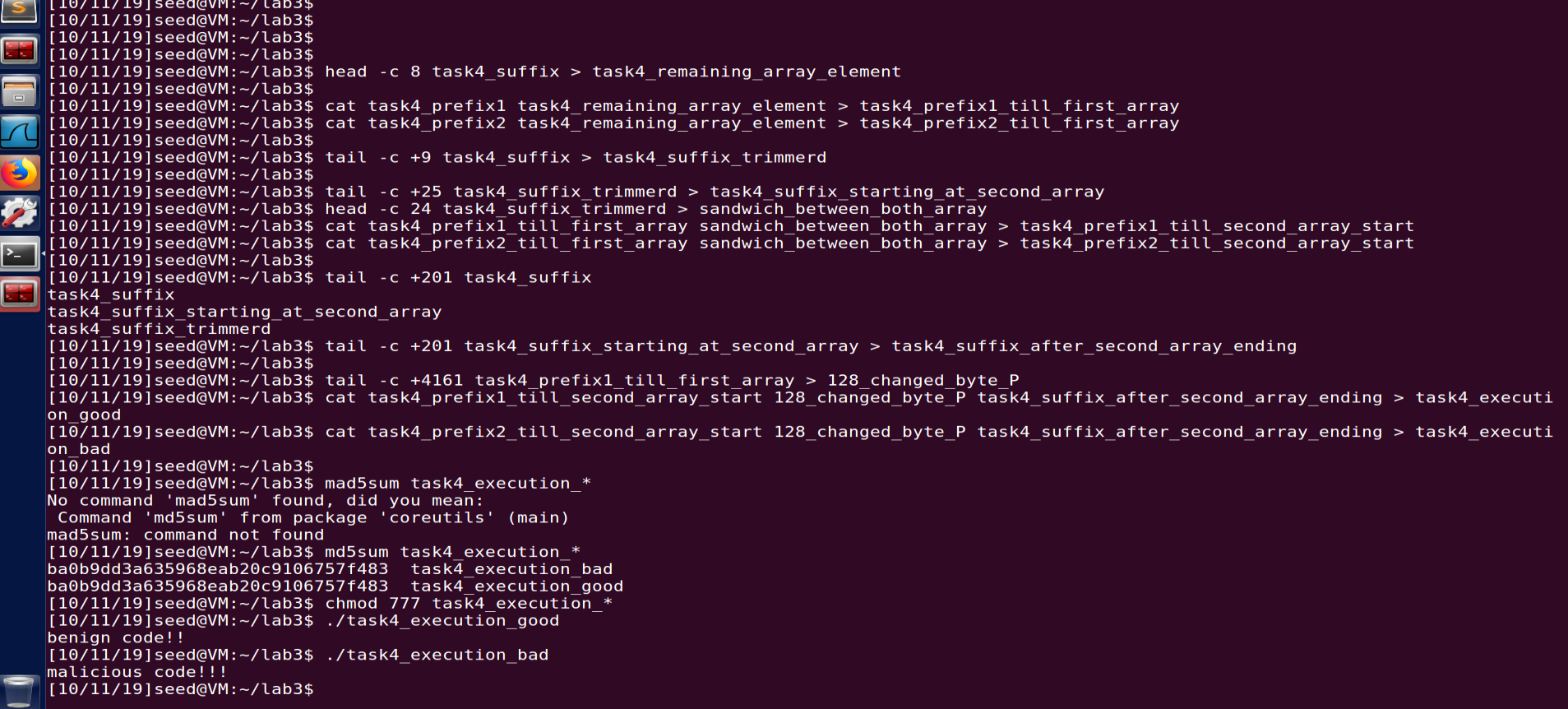
Let create bad and begin programs:

Good program will have those modified 128 bytes in array A and B (which will P & P according to guide) will bad will have Q and P.

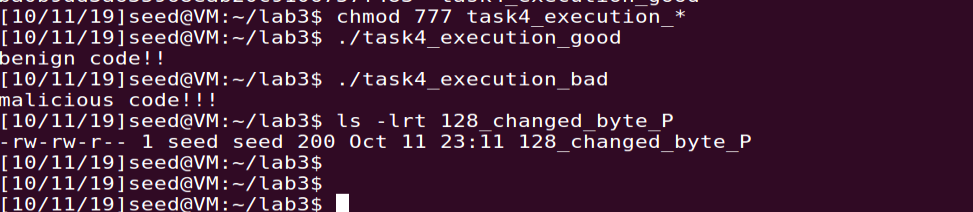
Now we need to extract 128 bytes from and copy in array B at right location in both good/bad program.

Step:

1. Get starting 8 byte, which is reaming element in array A in the program from the suffix.
2. Append these in both programs, so now both programs have full array A.
3. Remove those 8 bytes from suffix and create suffix which will be starting from end of array A to the end of program.
4. Now move get the byte between end of array A and starting of array B and append at the end of prefix in step 2. And remove those sandwich bytes from suffix
5. Now both program has code will starting of array B. and suffix has code till end.
6. Now remove 200 byte array B from suffix and create new suffix.
7. Now fetch 200 byte from program 1 ,which will have array A and we call in 128 \_changed\_byte array.
8. Now concatenate, code will starting of array B + 128\_changed\_byte\_array + suffix from end of array B to end of program.
9. Check Hash of both program. Which is same.
10. Run to see both program that they have different path.

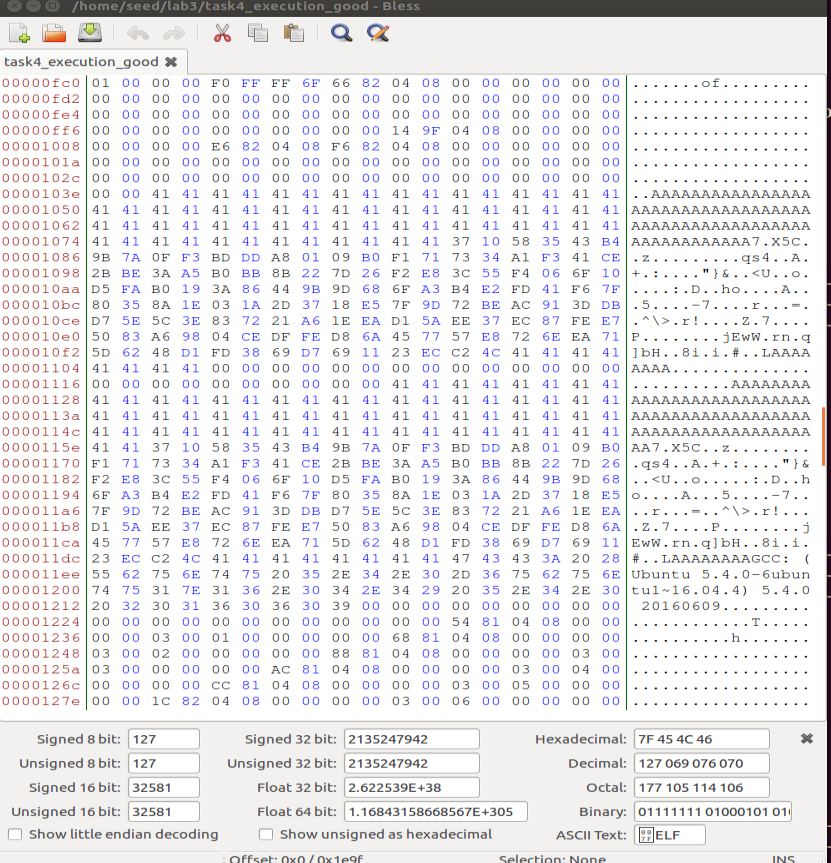


To avoid confusion with “128 changed byte” name: it is the array which has 128 byte changed but whole length is 200 byte.



**Bless of both program:**

Program 1:



**program 2.**

