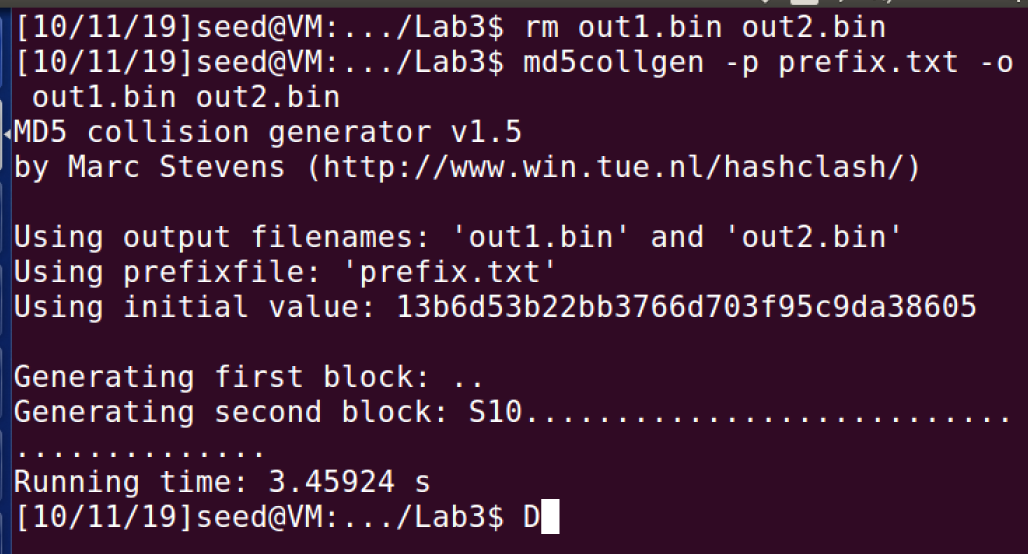
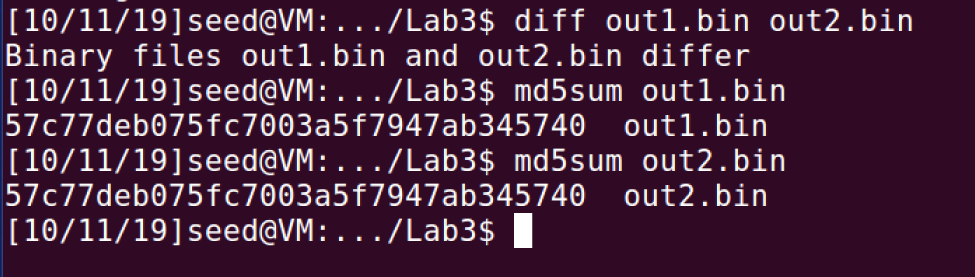
Lab 3

### CMPSC 443

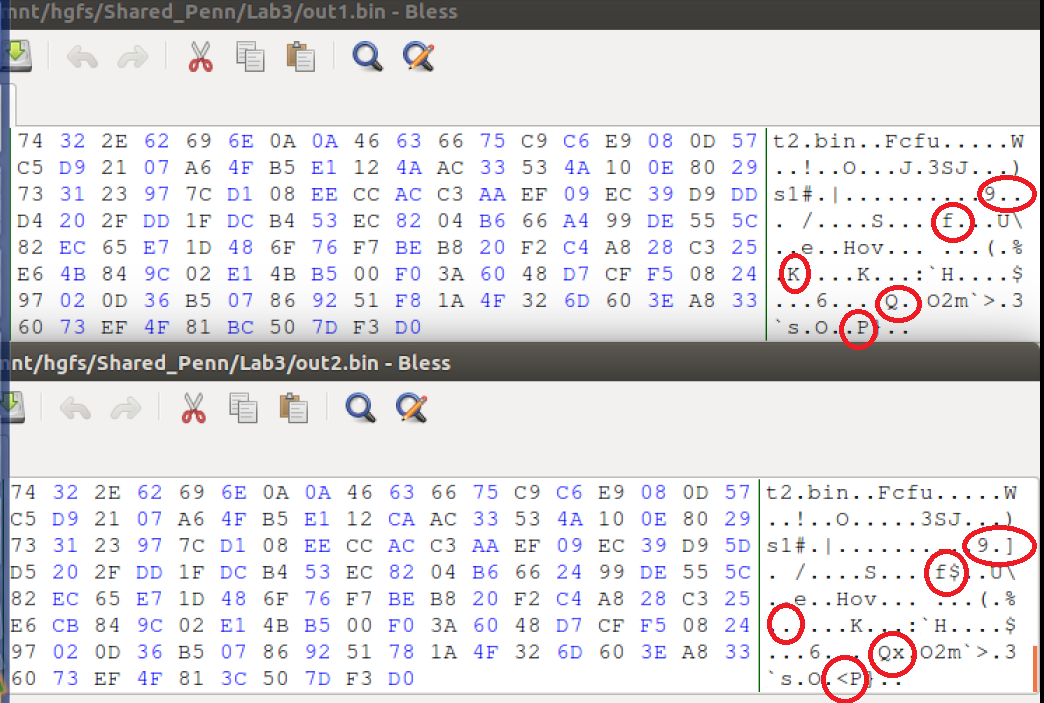
##### Student name:

# Task 1: Generating Two Different Files with the Same MD5 Hash





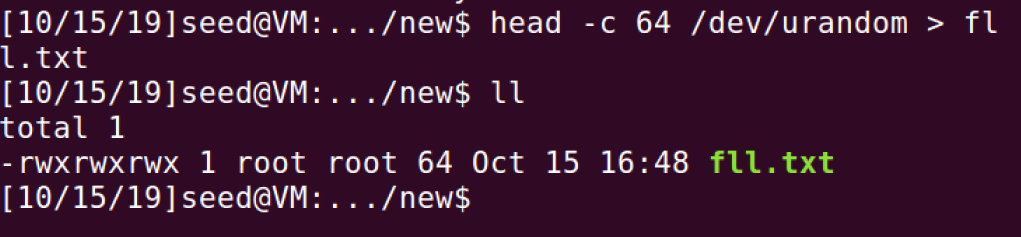
We can see the difference in the end of the bin files

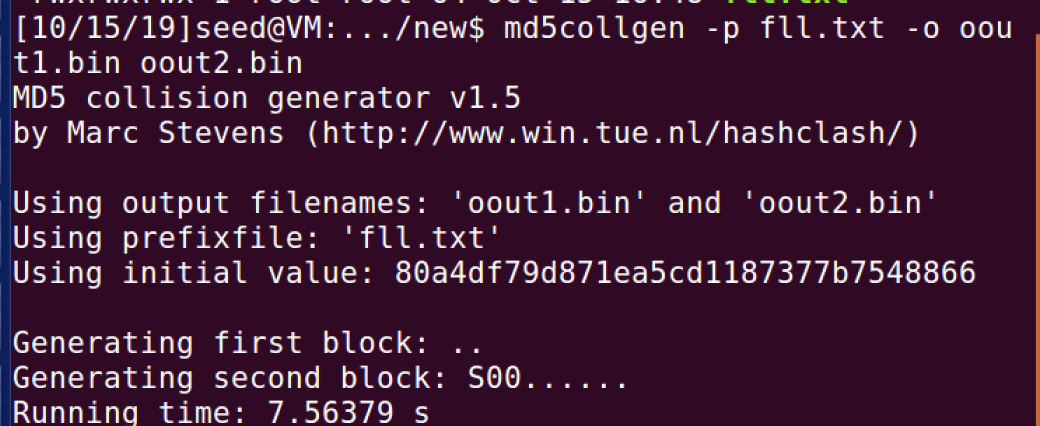


## Question 1

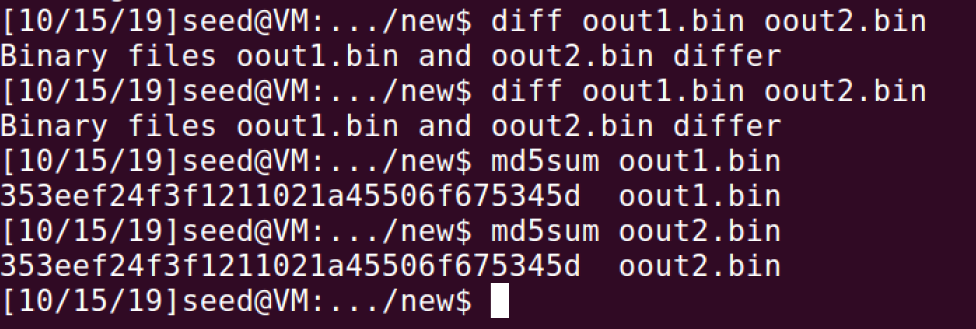
The file will be padded to be a multiple of 64

## Question 2



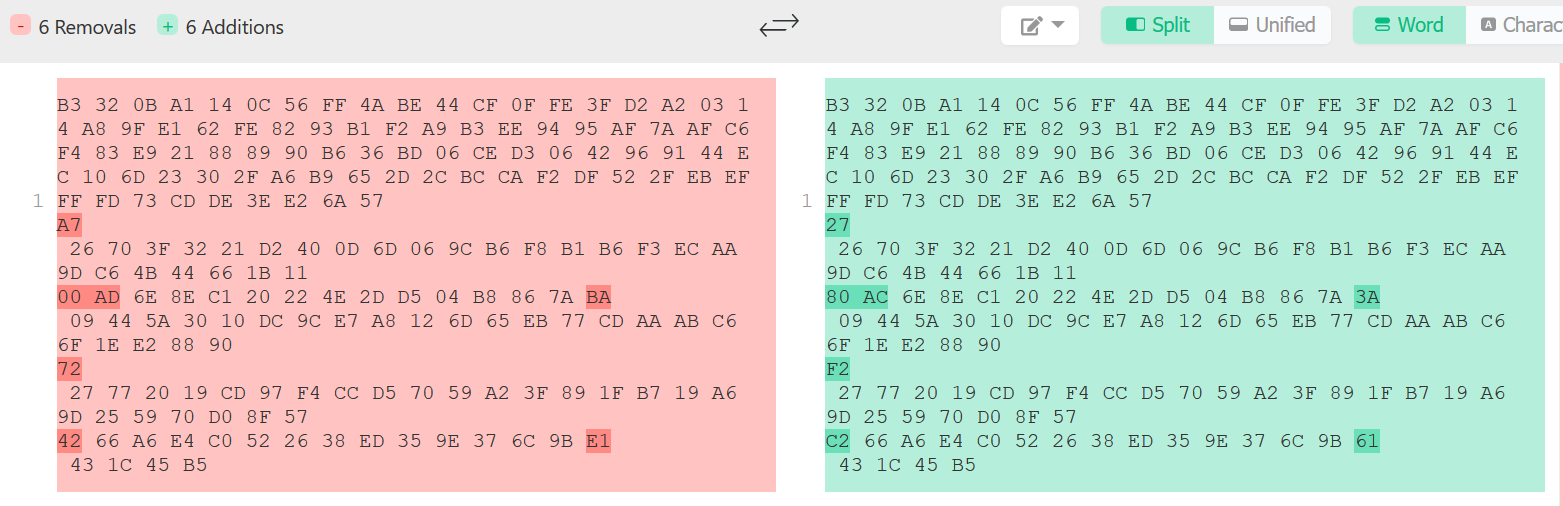


Now we see that the hashing is the same – because there is no padding



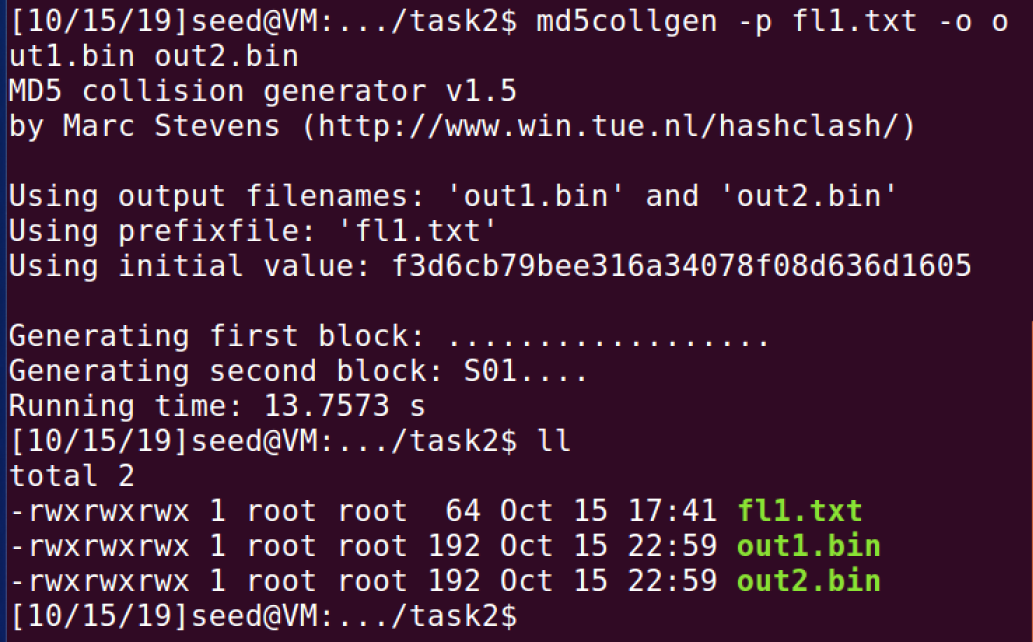
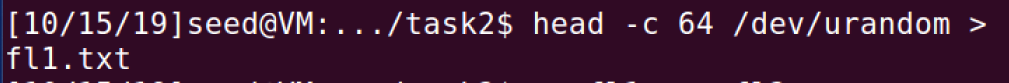
## Question 3

Not completely, but it is different in several places as shown below-

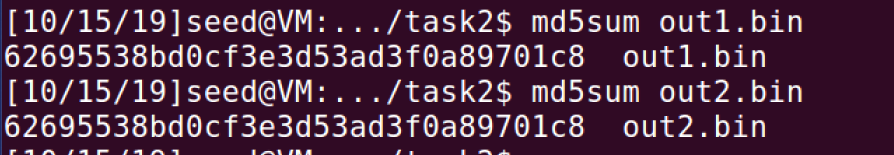


# Task 2: Understanding MD5’s Property

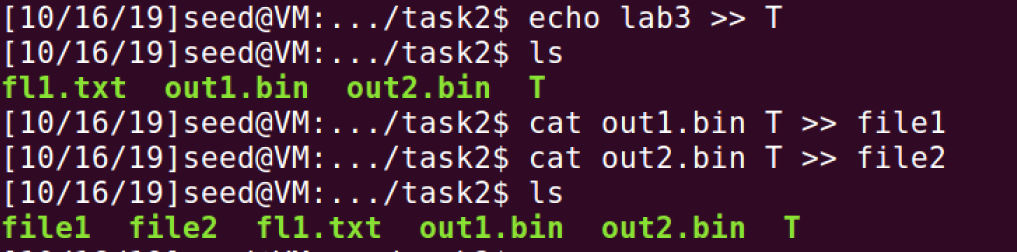
Creating the identical files

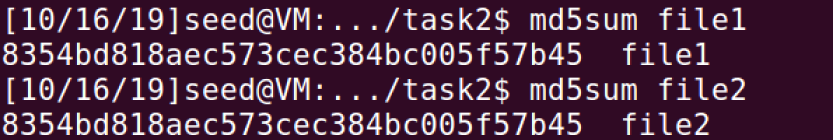


Making sure that MD5(out1.bin) = MD5(out2.bin)



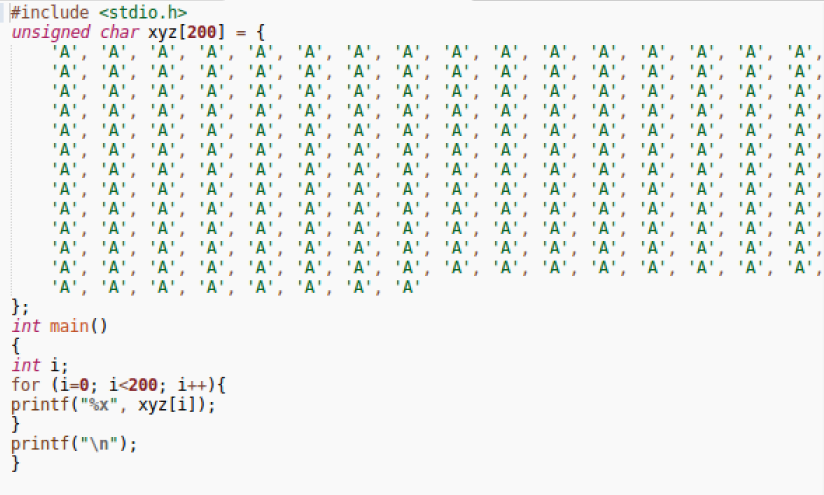
Now let’s add the “lab3” to file T, and concatenate T to out1.bin and out2.bin

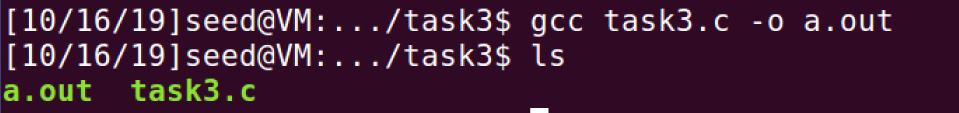


Now let’s show that MD5(out1.bin|| T) = MD5(out2.bin || T).  


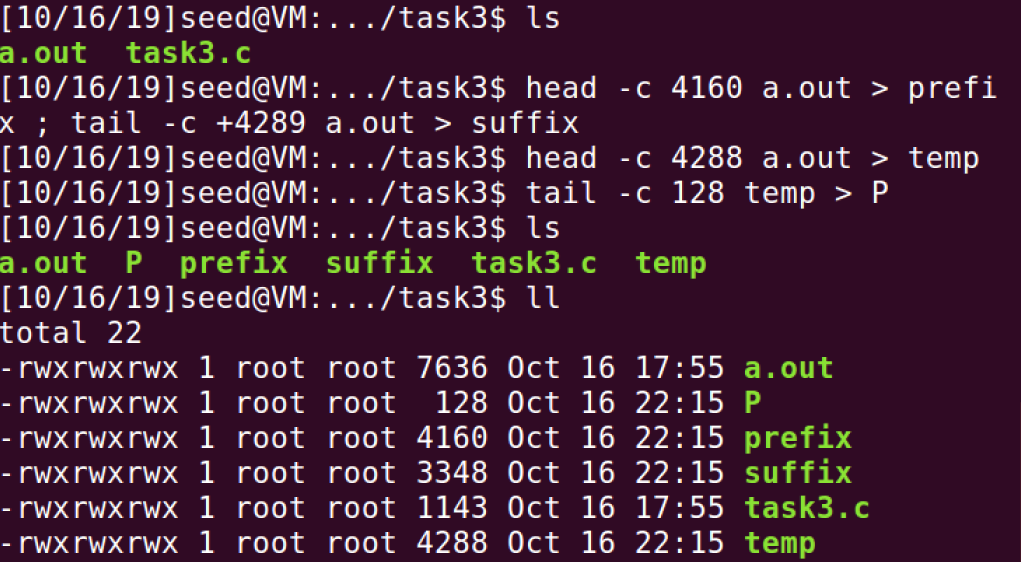
# Task 3: Generating Two Executable Files with the Same MD5 Hash

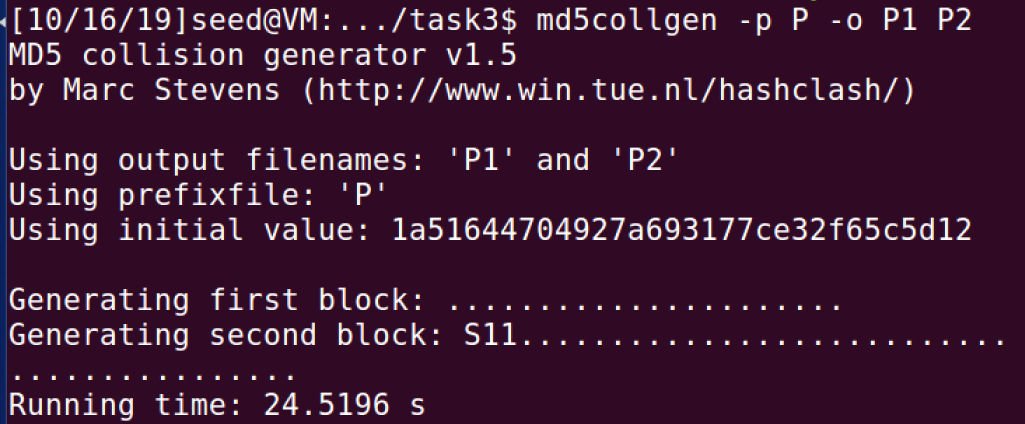
Make the first c file



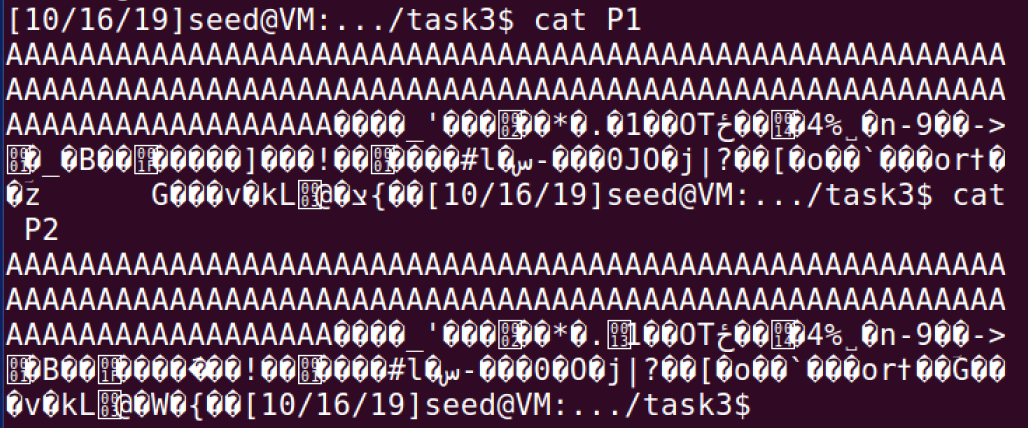


Making the different parts -   
the array starts at 4144, rounding up to a multiple of 64 we will get 4160.

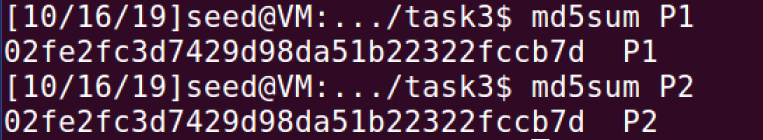


We will use `md5collgen` command to generate 2 file with the same hash value but with a different values.  


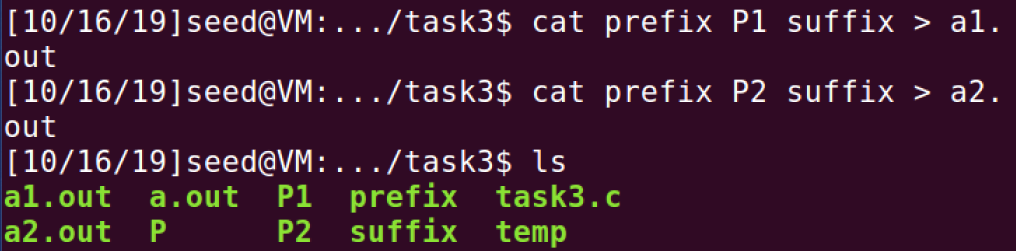
The different values:



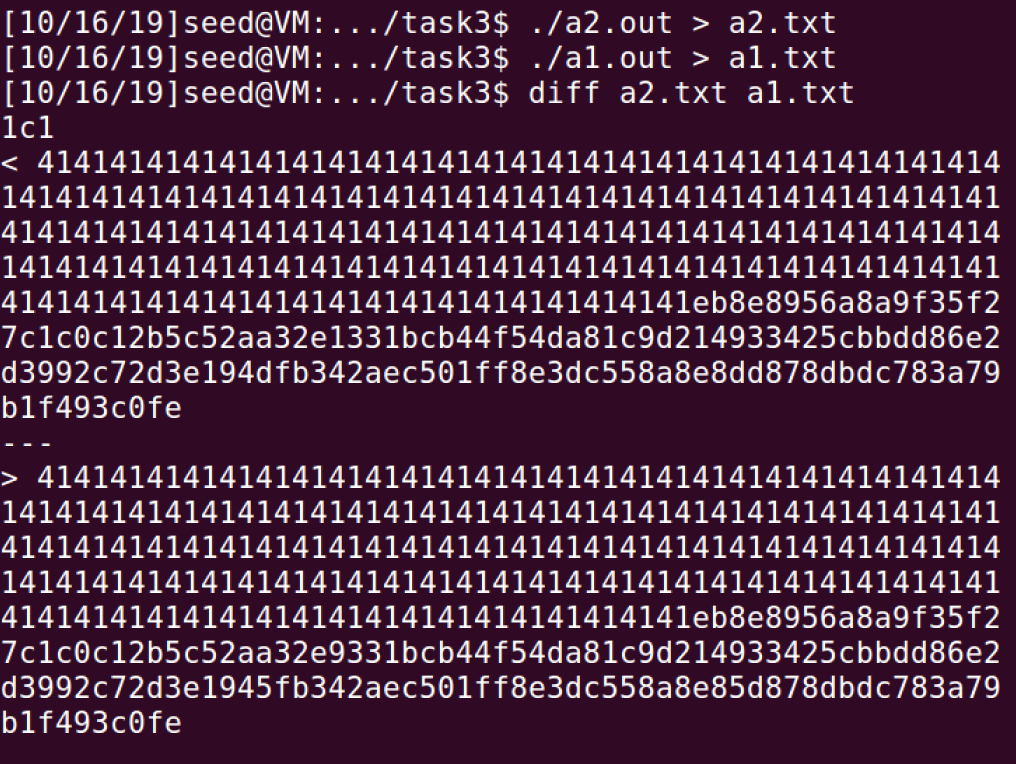
The same hash value:



Connecting the parts together:



Showing the different arrays with diff command:



# Task 4: Making the Two Programs Behave Differently

We wrote the following code in a c file

#include <stdio.h>

unsigned char xyz1[230] = { 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', …, 'A', 'A', 'A', 'A', 'A', 'A'};

unsigned char xyz2[230] = { 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', ... ,'A', 'A', 'A', 'A', 'A', 'A'};

int main(){

for (int i = 0; i < 230; ++i)

{

if (xyz1[i] != xyz2[i]) {

printf("Run `$ rm -rf /` command !!!\n");

return 0;

}

}

printf("I'm an innocent program o.O \n");

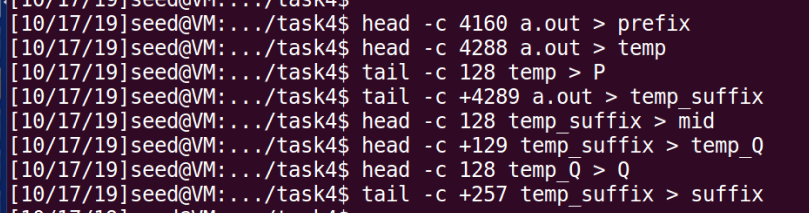
return 0;

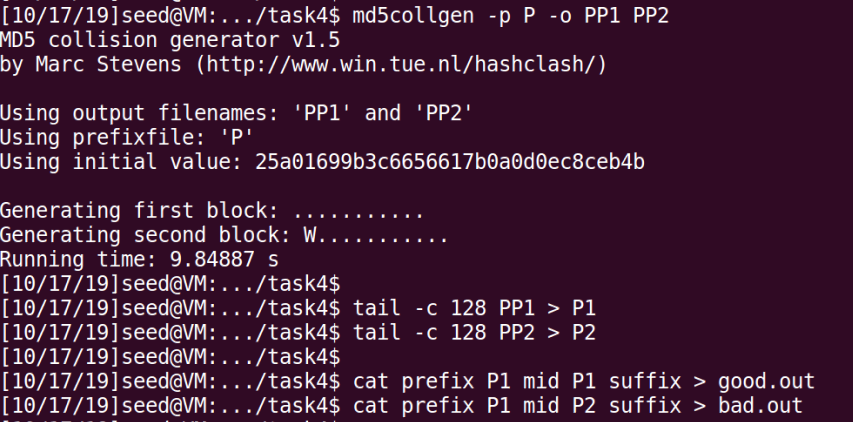
}

Now we will do the same thing we did in task 3



Commands to split the array to 5 parts (after checking the locations with bless program):





**Notice that we have cut out the original array after the m5dcollgen command and only left the 128 bits that were generated by the command, we use those 128 bits as P and Q (P and Q from the diagram)**

Now we will create 2 new programs with different arrays:

