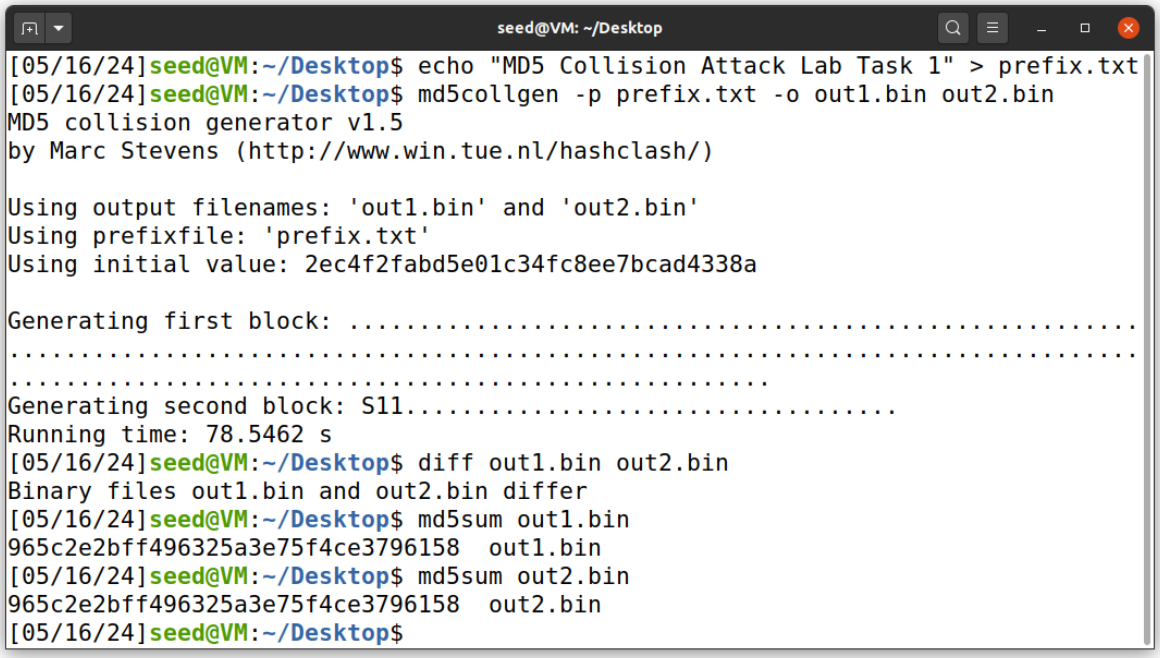
姓名：蔡佩蓉　　學號：109511286

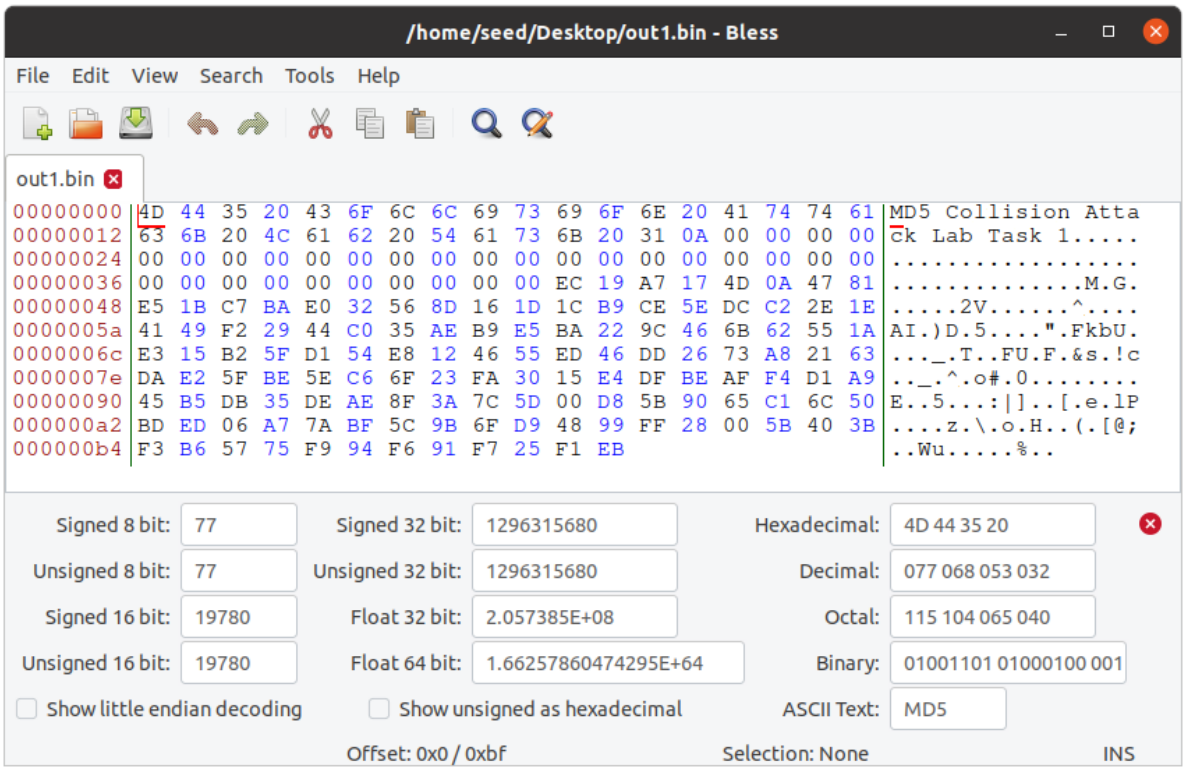
MD5 Collision Attack Lab (Lab4)

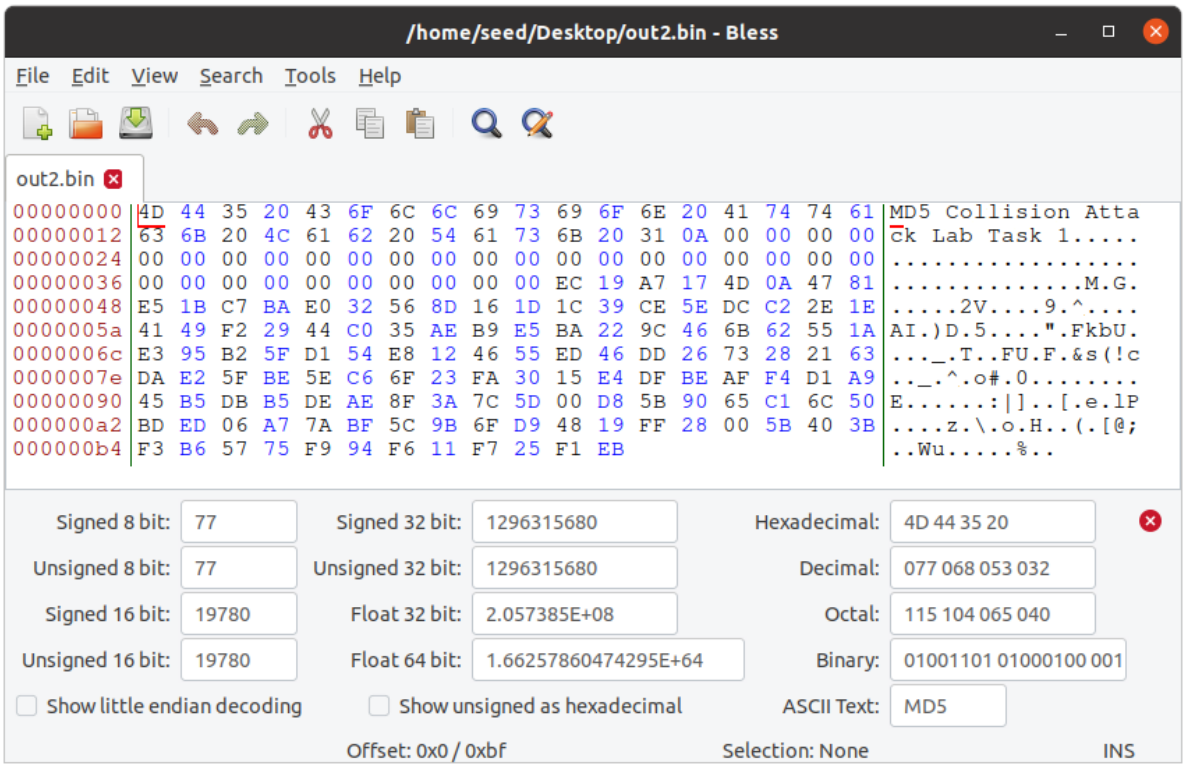
Task 1: Generating Two Different Files with the Same MD5 Hash

Questions:

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

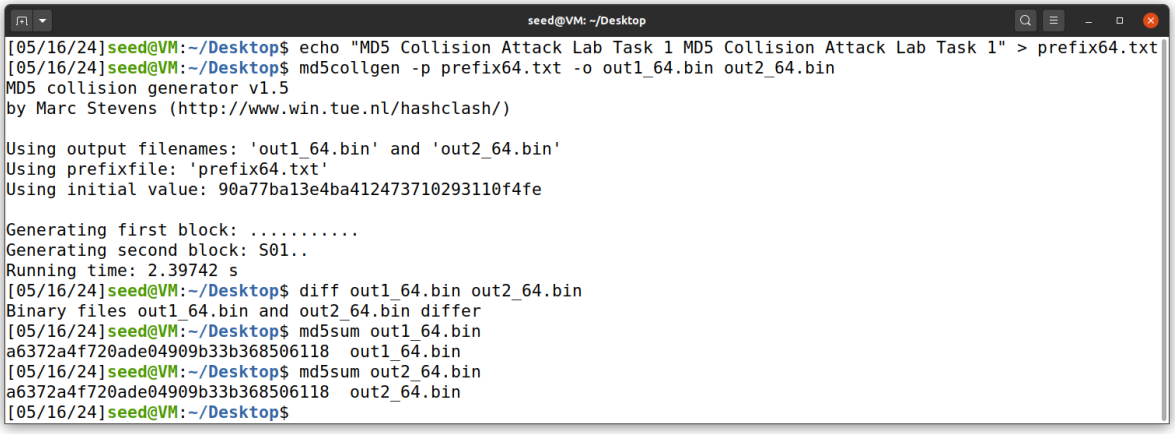


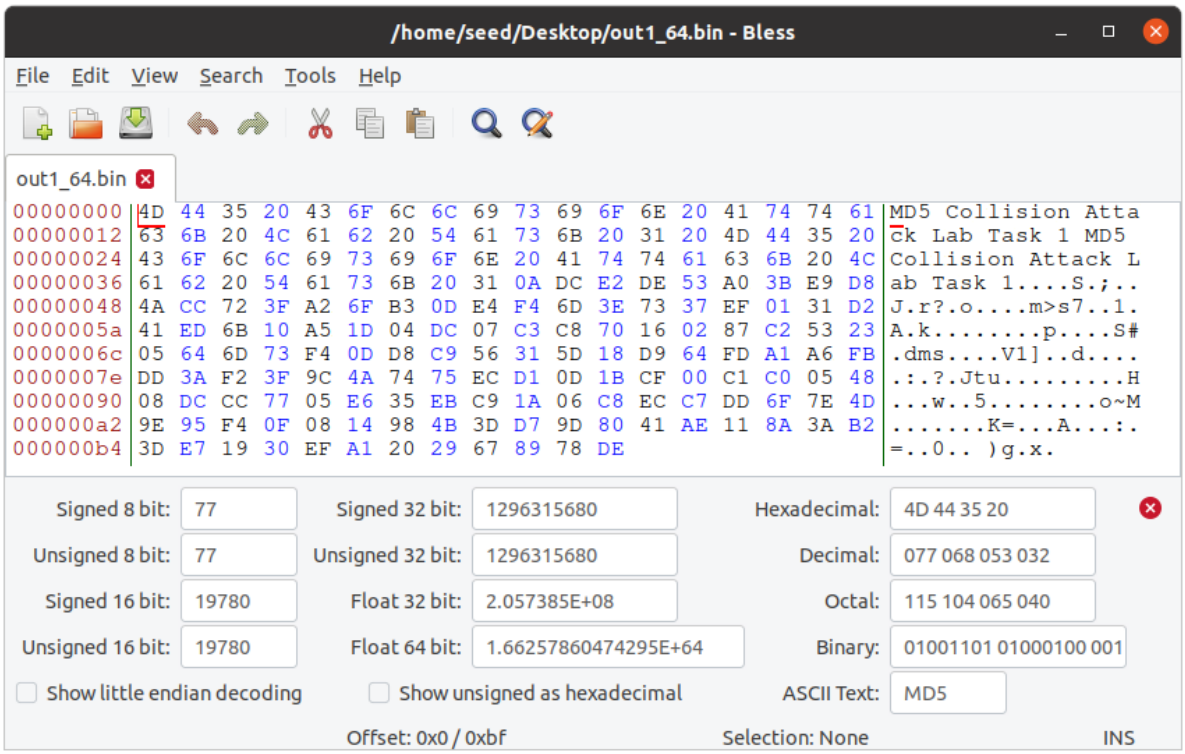


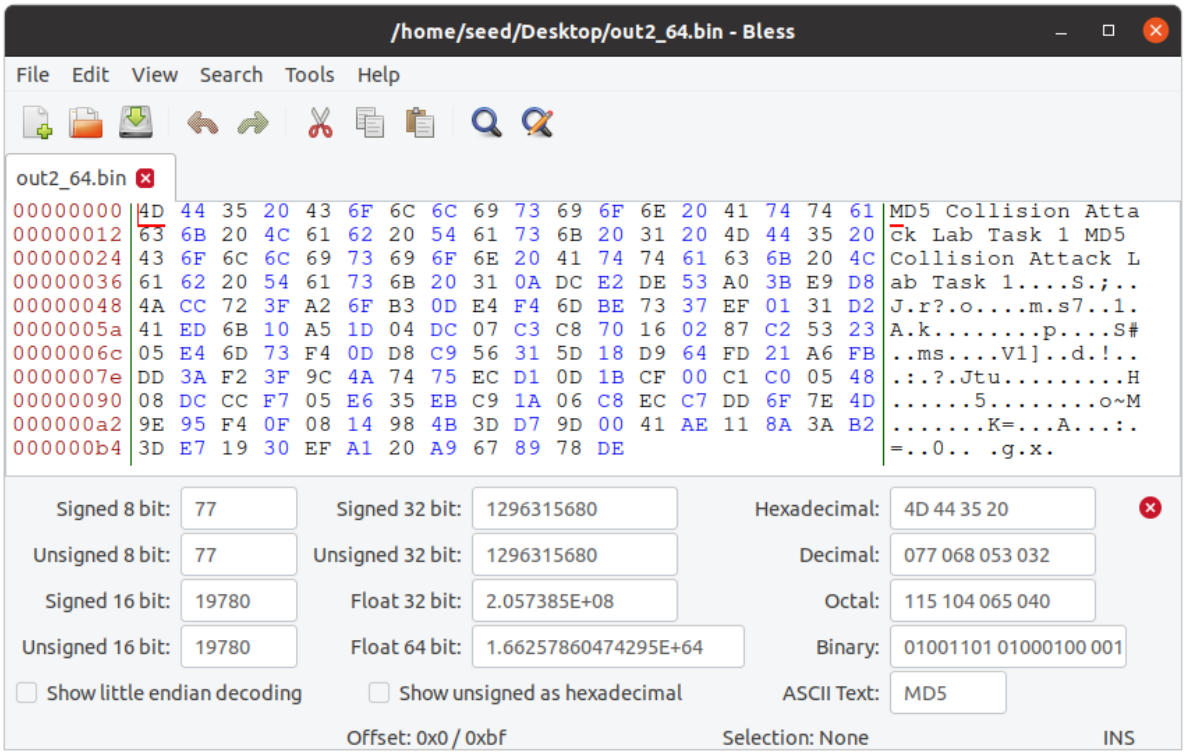


If the length of the prefix file is not a multiple of 64 bytes, the md5collgen tool will add padding to the prefix (padded with zeros). MD5 operates on 512-bit (64-byte) blocks, so the tool ensures that the prefix (plus padding) aligns to a 64-byte boundary before appending the collision blocks.

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

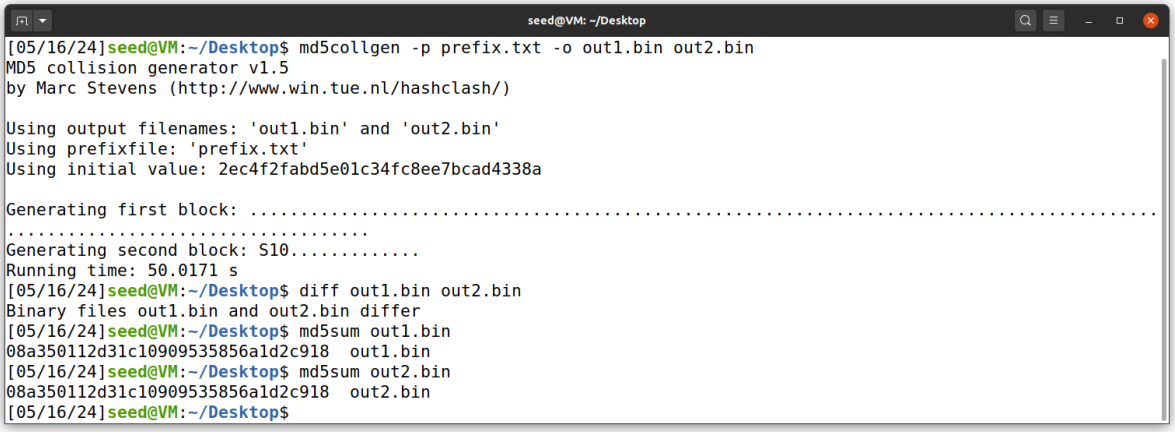


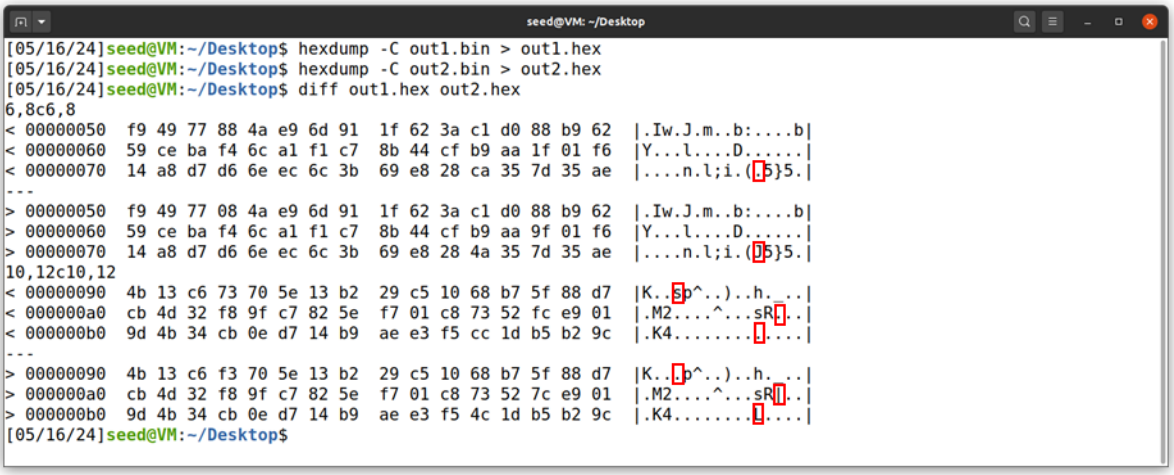


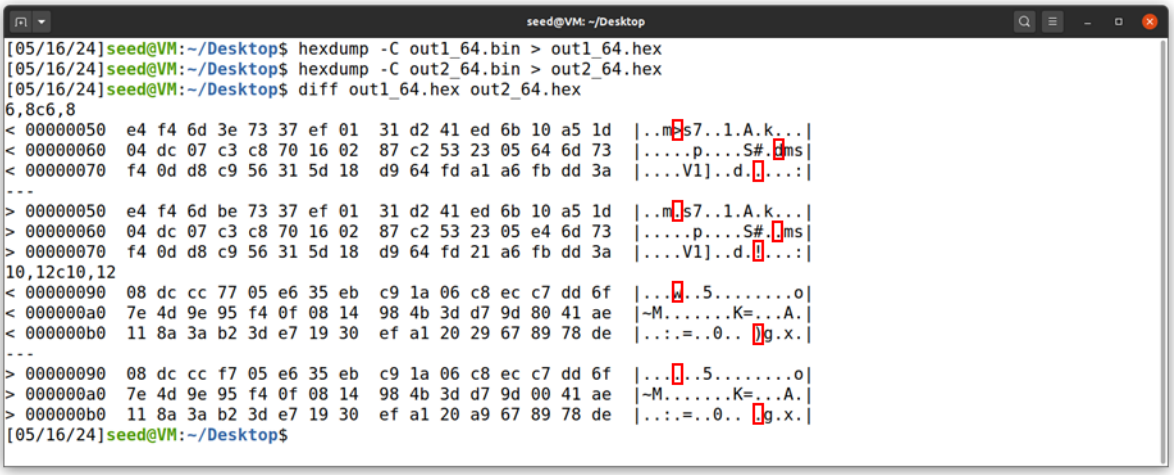


The md5collgen tool will not need to add any padding before appending the collision blocks (no zero padding). The two output files will still share the same 64-byte prefix, followed by their respective collision blocks.

3. **Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.**

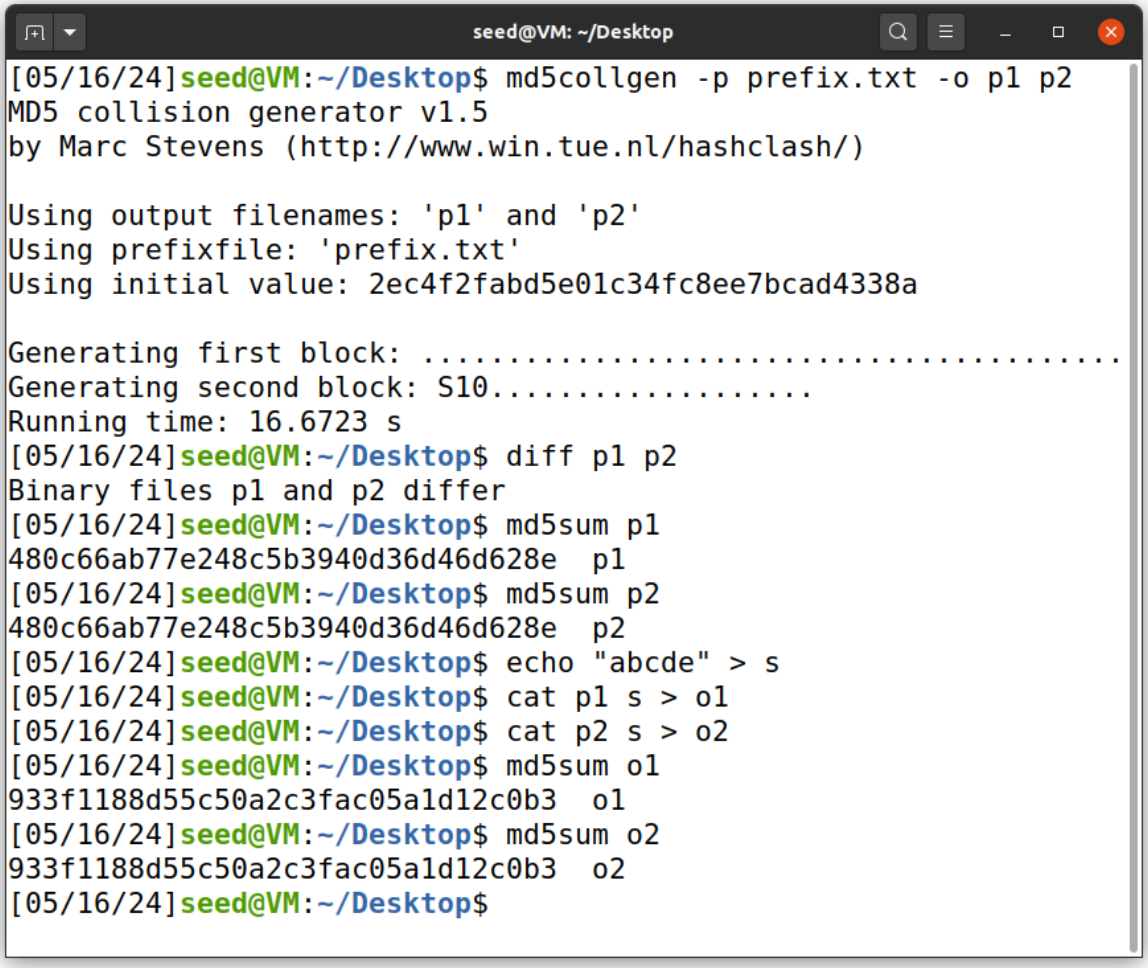






No, not all bytes are different (the different bytes are boxed above).

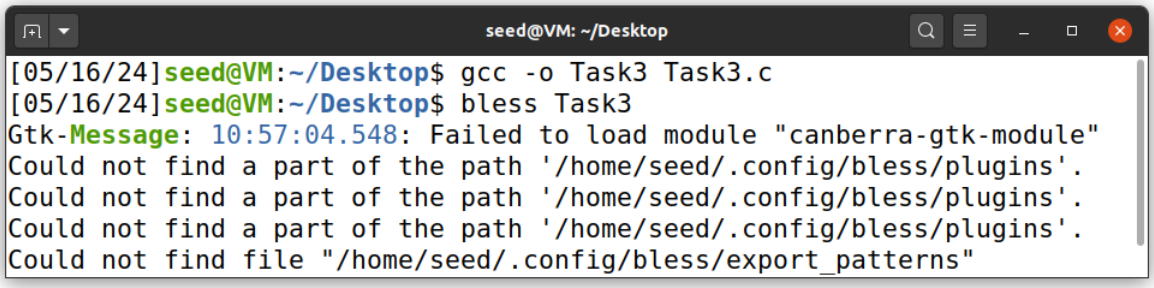
Task 2: Understanding MD5's Property

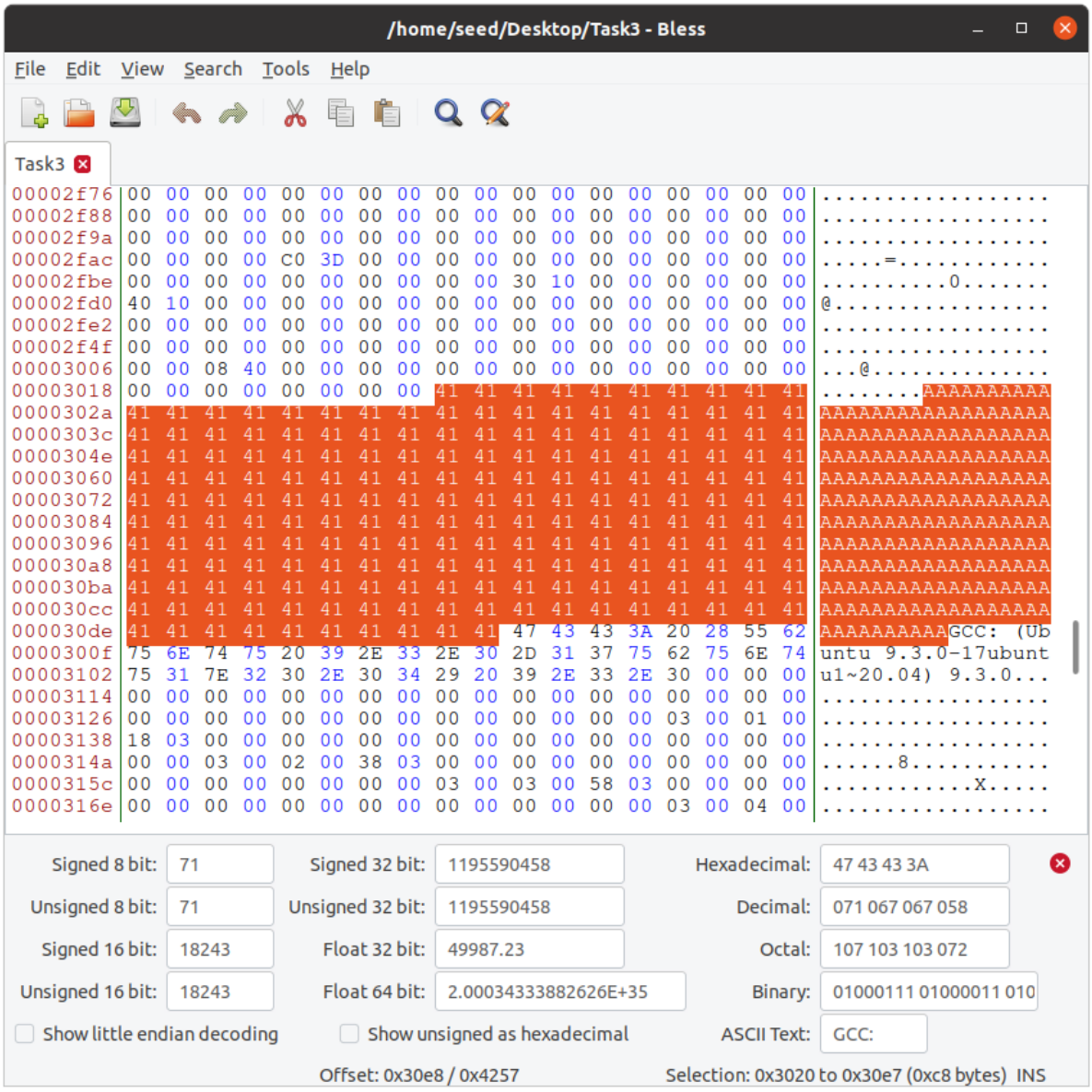


Property of MD5: appending the same suffix to two inputs that have the same MD5 hash will result in the same hash for the concatenated outputs.

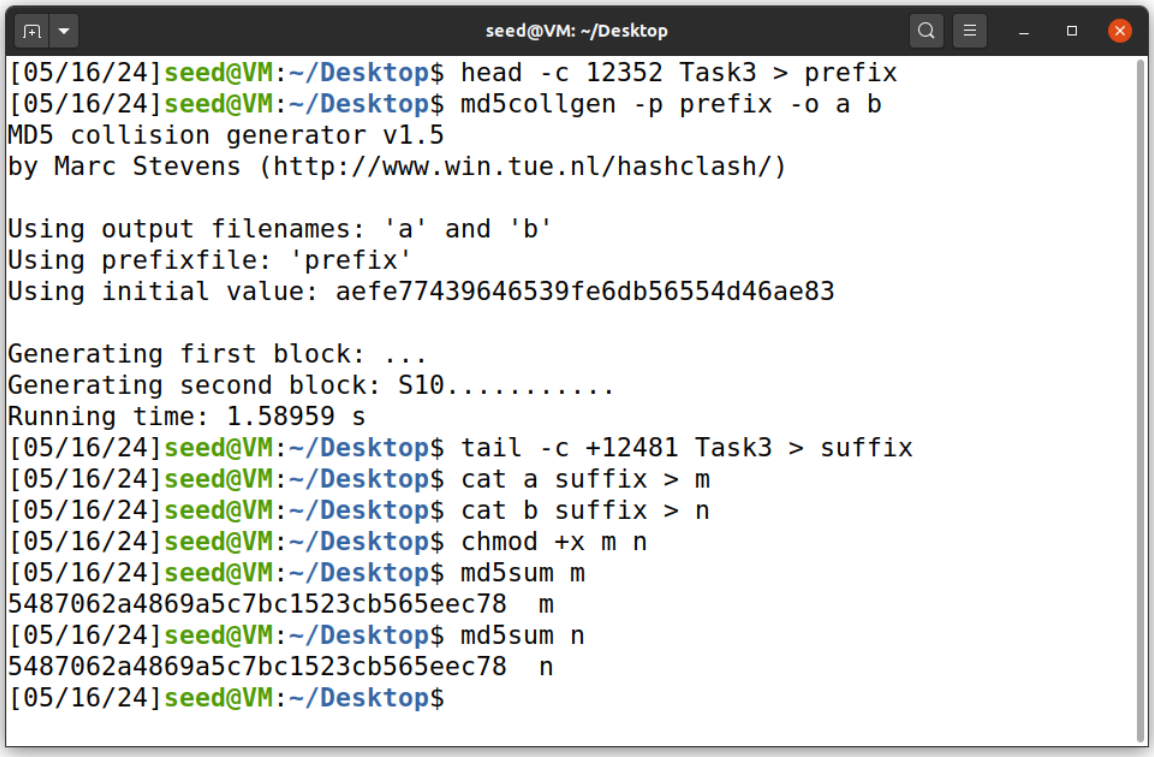
This is a fundamental property of many cryptographic hash functions and is crucial for understanding certain types of hash collisions and vulnerabilities.

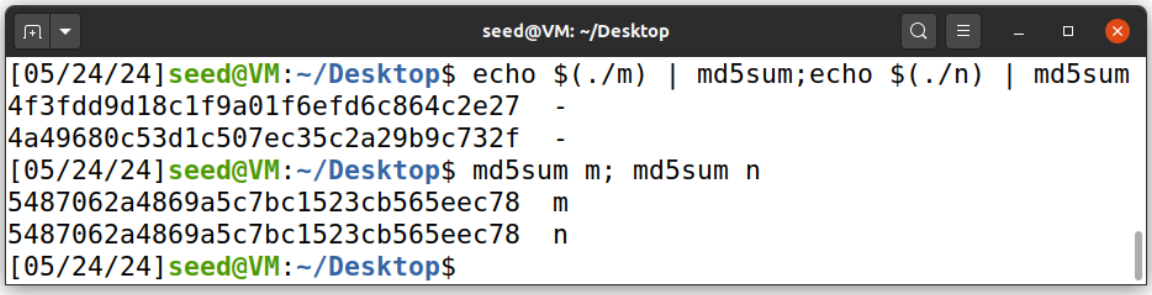
Task 3: Generating Two Executable Files with the Same MD5 Hash





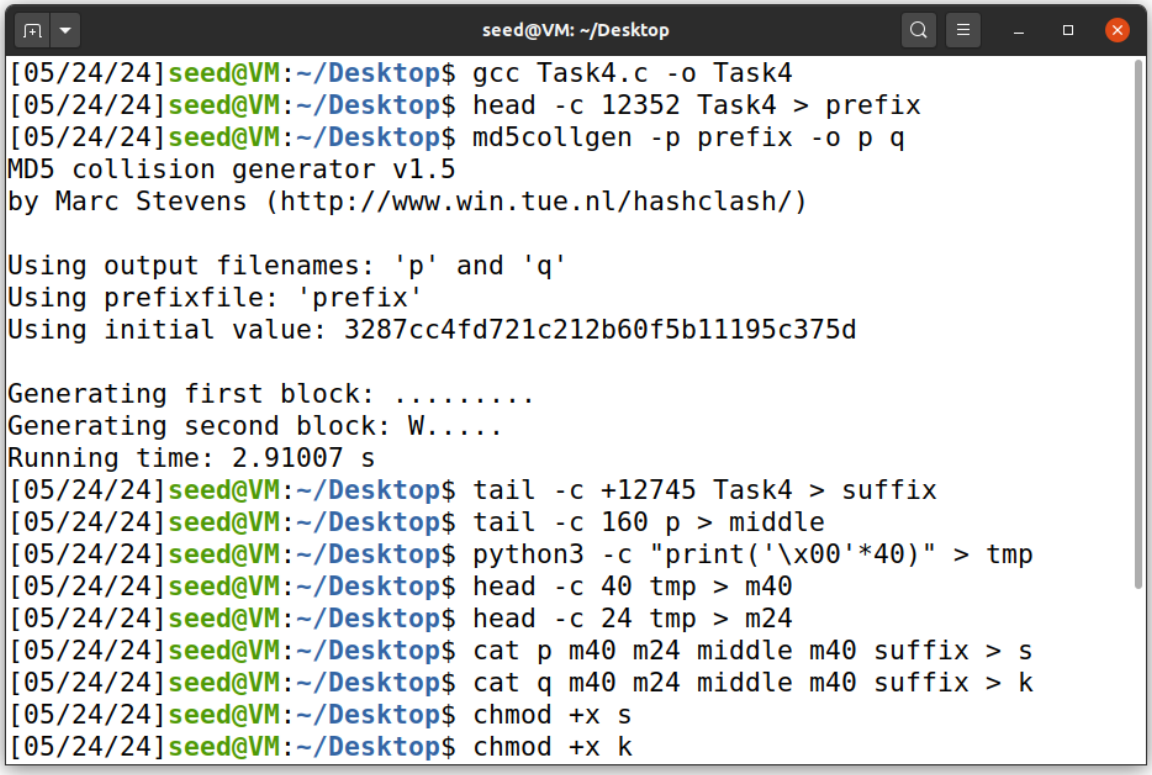
Length of prefix needs to be multiple of 64 bytes, 12320 (0x3020) is not multiple of 64, hence use 12352 (0x3040) [prefix = first 12352 bytes]. There is a 128-byte region, hence $12352 + 128 = 12480$ (0x30C0) as suffix [from 12481st byte to the end].



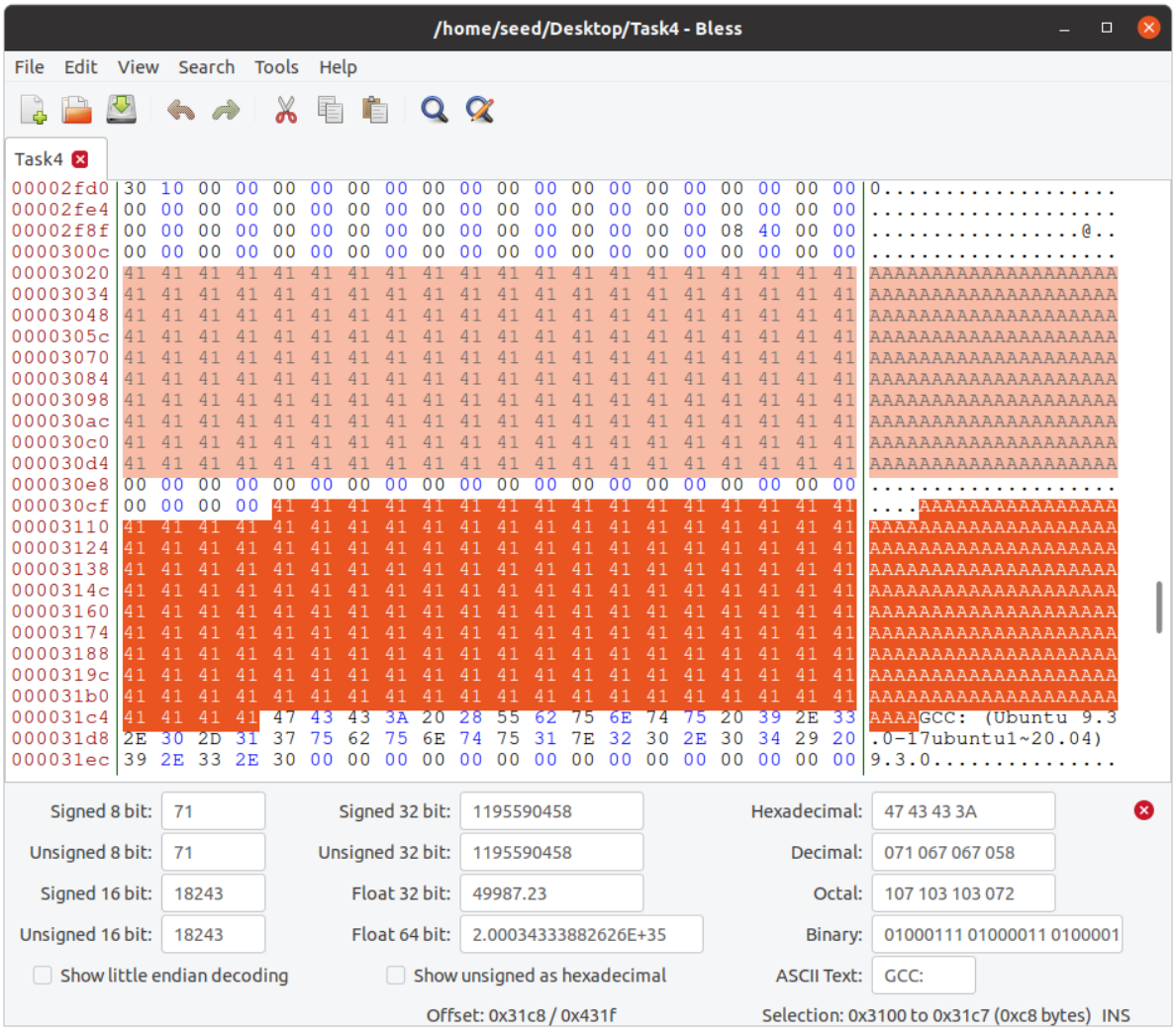


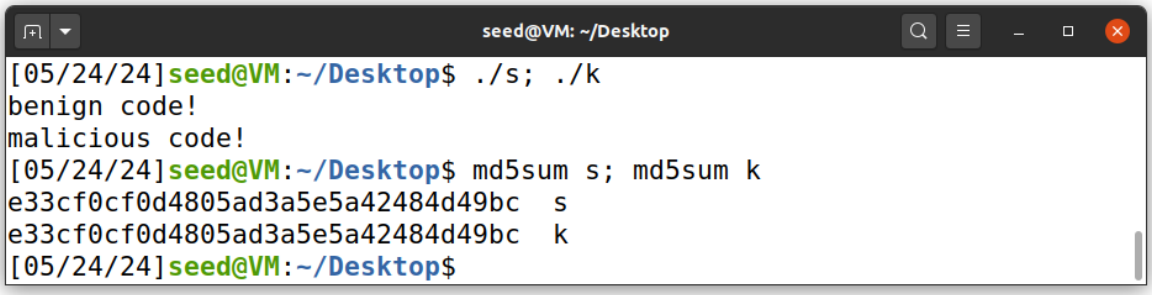
We observed that both have different outcomes (different content), but same hash value.

Task 4: Making the Two Programs Behave Differently



Get the last 160 bytes [32 (char A) + 128 (md5 padding)] from p. The length of the array is 200, hence, 40 bytes (200-160) need to be filled in to the a and b arrays. However, the difference between the two arrays is 0x3100 − 0x3020 = 0xE0 (224bytes), not 200 bytes, hence insert another 24 bytes of zero in between.





Executing both codes (s and k) returns different output, s returns benign code, k returns malicious code. However, the hash value for s and k are the same.