PROBLEM STATEMENT:

Suppose you have to store on any cloud service or on laptop that are confidential. You have to be sure that the files that you store are not tampered by any third party. You do it by computing “hash” of that file using a Cryptographic hash algorithm.  You save these hashes on your local machine.

Now, when you download the files, you compute the hash again. Then you match it with the previous hash computed. Therefore, you know whether your files were tampered or not. If anybody tamper with the file, the hash value of the file will definitely change. Tampering the file without changing the hash is nearly impossible

SOLUTION:

A hash is not ‘encryption’ – it cannot be decrypted back to the original text (it is a ‘one-way’ cryptographic function, and is a fixed size for any size of source text). This makes it suitable when it is appropriate to compare ‘hashed’ versions of texts, as opposed to decrypting the text to obtain the original version.

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available.This continues a series on bitwise operations and their applications, written by a non-expert, for non-experts. Follow [Biffures](https://medium.com/biffures" \t "_blank) for future updates. In SHA-256, messages up to 2⁶⁴ bit (2.3 exabytes, or 2.3 billion gigabytes) are transformed into digests of size 256 bits (32 bytes). For perspective, this means that an object [7 times the size of Facebook’s data warehouse in 2014](https://code.facebook.com/posts/229861827208629/scaling-the-facebook-data-warehouse-to-300-pb/) passed to SHA-256 would produce a chunk of data the size of a 32-letter string of ASCII characters, and that string would the object’s very special fingerprint.

Hash functions transform arbitrary large bit strings called messages, into small, fixed-length bit strings called message digests, such that digests identify the messagesthat produced them with a very high probability. Digests are in that sense [fingerprints](https://en.wikipedia.org/wiki/Fingerprint_(computing)): a function of the message, simple, yet complex enough that they allow identification of their message, with a very low probability that different messages will share the same digests.

A prominent use case of hashing is data integrity verification of large files, which relies on the comparison of actual and expected message digests, or [checksums](https://en.wikipedia.org/wiki/Checksum). Another is hashing as part of the encryption/decryption journey.

The operations included in the project are:

1. Add a new file hash value in the main disk file to track.

2.Check whether the newly downloaded file is tampered or not by checking with newly computed hash value and old value stored in the main disk file.

3. update the hash value as you yourself make changes in the file.

4.Delete the track of a file which no more exists or need not be tracked.

5. Insert all changes in the main disk file.