Prishi Amin
20998638

ECE 250 Project 2 Design Document

Class Design

1) <u>Fileblock Class:</u> The fileblock class manages the fileblock storing its ID, payload (as a char array) and the checksum. I kept the ID, checksum, and payload private to prevent their values from being manipulated by the user.

2) <u>Hashtable Class:</u> Handles how file blocks are stored and accessed. It uses separate chaining or open addressing to prevent collision. I used vectors to implement my hashtable (vector<vector<Fileblock*>>) as it allows dynamic resizing and simplifies how separate chaining is handled, where each slot in the hash table should be able to store multiple Fileblock pointers in a vector.

Function Design

1) <u>Fileblock Class:</u>

- **compute_checksum():** Uses the characters directly in its arithmetic operation because in C++ each char variable represents a character using its ASCII value in memory. Thus, this method gets the sum of the ASCII values of all the characters in the payload, and then takes modulo 256 of the sum to get the checksum value.
- **set_payload():** Sets a new payload for a file block instance and updates the checksum if required. It is used when corrupting the file block by updating the payload, but not recomputing its checksum.
- **compare_new_checksum():** Generates a new checksum and checks for corruption by comparing it with the existing checksum value.

<u>2) Hashtable Class:</u>

- **hash_func_1():** This creates the probe for the primary index of the key.
- **hash_func_2():** This creates the offset used for double hashing. It will always return an odd number.

Runtime Analysis

STORE: For separate chaining, we can directly insert it into the vector at the index calculated by

the hash function which is O(1) time. For double hashing, only a constant number of positions are probed to find an empty slot using the offset calculated by the second hash function, which is also O(1) time.

SEARCH: For separate chaining, we use the first hash function to find the chain that our index will belong to, which takes O(1) time. For double hashing, only a constant number of positions are probed to find the File block, which is also O(1) time.

DELETE: For separate chaining, we can directly find the required chain and remove the File block, which takes O(1) time. For double hashing, the same sequence to find the index as SEARCH is used, which makes DELETE also have O(1) time.

UML Diagram