4. Test Cases

It is important to test edge cases to confirm all functions are working as expected.

Open Addressing:

- Capacity Test: Insert a new key into the full table
- Collision Test: Insert a key that the result of primary hashing gives collision (secondary hash function should be called and find the appropriate index to insert the key)
- Duplication Test: Insert a key that is already in the table
- Deletion Test: Delete a key that is in the table or not in the table
- Search Test: Search for a key that is in the table, not in the table or deleted from the table

Separate Chaining:

- Duplication Test: Insert a key that is already in the table
- Collision Test: Insert a key that the result of primary hashing gives collision (key should be inserted into the chain in descending order)
- Deletion Test: Delete a key that is in the table or not in the table
- Search Test: Search for a key that is in the table, not in the table or deleted from the table
- Print Test: Print a key from an empty chain / Print to check if the keys are ordered in a descending order

5. Performance

Commands with a time complexity of O(1) are insert, search, and delete. A command “insert” inserts a new key into the hash table using the hash function. For both open addressing and separate chaining, assuming uniform hashing, finding the appropriate position to insert can be done in O(1), and inserting key/value also has a time complexity of O(1). Hence the resulting time complexity is O(1). A command “search” searches a key from the hash table by utilizing the hash function. Assuming uniform hashing, a searching operation can be done in O(1) for both open addressing and separate chaining. Finally, the command “delete” deletes a key from the hash table if it exists. Since it behaves like a “search” except for deleting the found key from the table, considering that deleting the key is executed with the time complexity of O(1), deleting operation can also achieve a time complexity of O(1).

Command "print" can’t achieve a constant time complexity since it needs to traverse the whole chain to print out all the keys. Hence, assuming uniform hashing, the length of each chain will be n/m. Then, the time complexity would be O(n/m).