Group 20

I. Contribution:

Thu Nguyen: worked on main function, wrote the report

Huy Huynh: worked on main function, did unit tests

Linh Tran: worked on HashTable1D and HashTable2D classes

II. Instructions for Compiling and Running the Program:

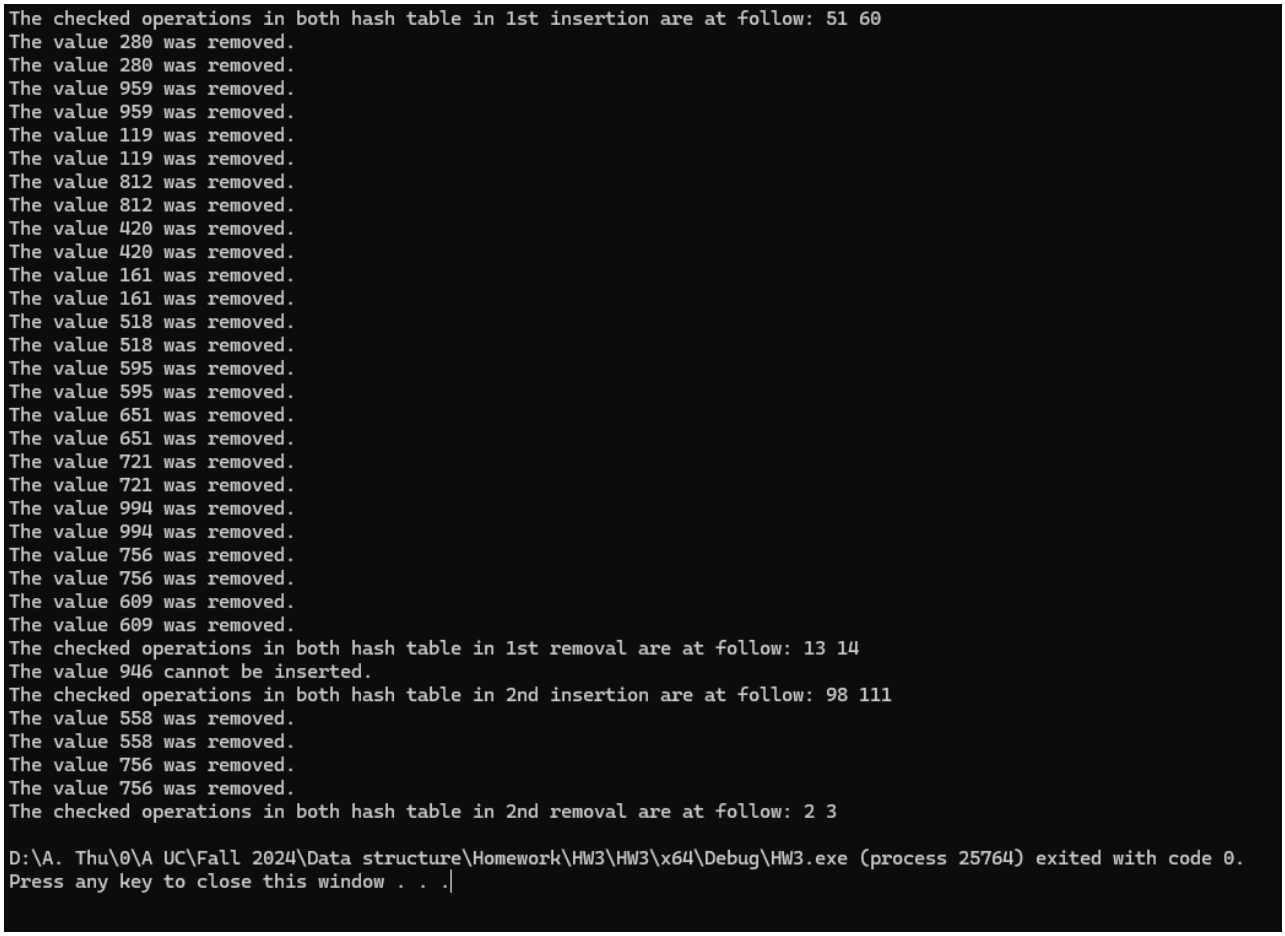
- **Required Files**:
  - HashTable1D.h and HashTable1D.cpp (for the first hash table functionality)
  - HashTable2D.h and HashTable2D.cpp (for the second hash table functionality)
  - HW3.cpp (main program logic)
  - UnitTest1.cpp and UnitTest2.cpp (for the unit test)
- **Compiling**:
  - Go to **File > New > Project** and create a new **Console App**
  - Add each .h and .cpp file by right clicking the project name in the **Solution Explorer** and selecting **Add > Existing Item...**.
  - For the unit test, the path for reference file may varies depend on users
  - Go to **Build > Build Solution** or press Ctrl + Shift + B. (for the main program)
  - For unit test, change the path location if needed, go to **Test > Test Explorer > Run all test.**
- **Running the Program**:
  - After building, click **Debug > Start Without Debugging** or press Ctrl + F5 to run the program.

III. Results analysis:

According to the test result, we can see that the first hash table requires less spots checked than the second one in both the insert and remove process. It meets our expectation, and it is undeniable that the first harsh table works better.

For the first hash table, the best ideal situation is with a low load factor and good hash function, but the performance decreases if it works with high load factor and poor hash function.

For the second hash table can handle moderate load factors well but becomes less efficient when buckets overflow due to high load factors or poor hash distribution.



In this assignment, we utilize AI in debugging the unit test (the unresolved external symbol error) .