**// In main():**

// Ruichun Chen: main(), menu(), compare(), readFile(), dataValid(), leverOrder(), inOrder()

// Xin Jing: insert(), remove(), findNode(), zipCode pointer, database class, vector, code integration and debug

// Wenhui Li: constructor, \_getPrime(), \_isPrime(), \_rehash(), insertHashTable(), searchHashTable(), printHashTable(), deleteHashTable(), code integration and debug

// Wenzhe Xu: goodhash(), badhash(), destrcutor, destroyHashTable(), fomatting

// De Yi Huang: part of file I/O, structure design

// The main funciton contains a database class:

// The class of Database:

// 1,Four member variables: Two BinarySearchTree tree Zip1, Zip2, One hashTable HashZip and a Stack for undo function.

// 2, Class saveNode and save function: saveNode a function object and used for the output file. Save function use the saveNode function object to implement the file output and clear the stack.

// 3, The constructor of the database class is to load the input data from input file and build the tree and hashTable.

// 4,The destructor is to clear the shared Zip code pointer in the heap and clear the pointer in the stack.

// 5,UndoDelete function: undo the delete in the reverse order of the delete sequence

// 6, deleteZipCode function : delete the tree and hashtable by zipcode and store the delete item in stack

// 7, deleteCityName function: delete the tree and hastable by cityName and store the delete item in stack

**// In BinaryNode:**

// BinaryNode uses to save the Zipcode pointer in item, and it has the left and right pointer which points to the next BinaryNode

**// In BinaryTree:**

// BinaryTree can do the traverse the code from tree, print the zipCode pointer inOrder, preOrder, postOrder, breathOrder, levelOrder, insert, delete, getEntry

// 1,inOrder function: use to zipCode \* to get the zipCode and cityName in the tree, and print out the data in zipCode and cityName sort in ascending order

// 2, levelOrder function: the function is used to print out the indented tree, by using the Queue

**// In BinarySearchTree:**

// BinarySearchTree inheritances from BinaryTree, and do the insert, delete, remove,removLeftMostNode, findNode, findSmallest, findLargest.

// 1,insert function: use to insert the zipCode \* into the tree. Add the function pointer into this function.

// 2, remove function: the original code would not workable for the pointer because compare two pointer. Use the function pointer to make the compare workable.

// 3, getEntry function: it used to find the target item and store the item in vector. As it may have duplicate items and we do not know how many it is. We prefer to use the vector to store them.

**// In ListNode:**

// ListNode uses to save the Zipcode pointer in item, and it has the next pointer which points to the next ListNode

**// In LinkedList:**

// LinkedList can do the traverse the data from List, and print the data in the same bucket

// 1,traverseList function: use to zipCode \* to get the zipCode in the LinkedList

// 2, searchList function: the function is used to search the data by the input zipCode

**// In HashTable:**

// HashTable has the LinkedList, and do the rehash, getTableSize, getBucketCount, getCollision, getLoadFactor, insertHashTable, deleteHashTable, searchHashTable, printHashTable.

// 1,insertHashTable function: use to insert the zipCode \* and update the bucket number and collision number

// 2, deleteHashTable function: this function delete the item from hashtable and update the bucket number and collision number.

// 3, searchHashTable function: search the item from hashTable.

**// In ZipCode:**

// 1. the zipCode class contains zipCode as primary key, cityName as secondary key, county name, total polulation, the polulation of Hispanic Latino, Asian, White, African American

// 2. the overloading > and < function, use the compare function to compare two values

// 3. the overloading the << funciton is used to print out the data

**// In Stack:**

// Stack is used to implement the undo delete function in main

**// In Queue:**

// Queue is used to implement the BinarySearchTree inOrder traverse.