```
//*****************************************************************************************/
// Exercise 3: Text2Binary & Hashing with Linear Probing     WuYH@ICE.CYCU
//*****************************************************************************************/
int main(void)
{    vector<studentType>     cSet;              // set of output records
     string                  fileID;            // file identifier
      string                   quitOrNot;       // stop the program or not
      int                     stNo;              // number of students

     do
     {    if ((stNo = Text2Binary(fileID)) == 0)    // transform a text file into a binary file
               return 0;                            // stop the program if nothing is obtained
          if (!getBinRecords(stNo, fileID, cSet)) // get records from a binary file
               return 0;                            // stop the program if nothing is obtained
          try
          {    int              htSize = leastPrime((float)cSet.size() * 1.2);
                               // hash table size is the minimum prime > (data size) * 1.2
                    hashTable     htObj(htSize);// initialize an object of hash table

               if (htObj.existed())                 // if a hash table exits, ...
               {    for (int i = 0; i < cSet.size(); i++)
                         htObj.insertOne(cSet[i]); // insert each record into hash table
                    htObj.outputAll(fileID);         // output hash table on the screen & in a file
                    htObj.failedCompAVG();// average number of comparisons for unsuccessful search
                    htObj.successCompAVG();   // average number of comparisons for successful search
                    htObj.clearAll();                // release the space allocated to the hash table
               }    // end inner-if
          }   // end try
          catch (std::bad_alloc& ba)                // unable to allocate dynamic space
          {    std::cerr << endl << "bad_alloc caught: " << ba.what() << endl;
          }    // end catch
          cSet.clear();                             // release the space of all records
          cin >> quitOrNot;
          if (!quitOrNot.compare("0"))              // press 0 to stop execution
               return 0;
     } while(true);    // end do-while
}     // end main
```