```
Proj4:
Class IRS
   Private:
      init hash table
      init int insCnt
      init int dltCnt
      init int rtrvCnt

   Public:
      Constructor IRS()
         init table with size 10007
         Set insCnt, dltCnt, rtrvCnt to 0

      Method ins(SSN: string, name: string)
         If table.insert(SSN, name) return true
            +1 insCnt

      Method rmv(SSN: string)
         If table.erase(SSN) returns true
            +1 dltCnt

      Method rtrv(SSN: string)
         If table.find(SSN) returns true
            +1 rtrvCnt

      Method printStats(elapsedTime: double)
         Output "The Number of Valid Insertion: " + insCnt
         Output "The Number of Valid Deletion: " + dltCnt
         Output "The Number of Valid Retrieval: " + rtrvCnt
         Output "Item numbers in the list: " + table.getSize()
         Output "Time elapsed: " + elapsedTime + " seconds"

Main func
   Start clock and set duration
   Create IRS object irs
   Open file argv[1] as input
   Declare firstChar, SSN, firstName, lastName as string

   While file has data
      Read firstChar, SSN, firstName, lastName
      If firstChar is "i"
           irs.ins(SSN, firstName + " " + lastName)
      Else if firstChar is "d"
         irs.rmv(SSN)
```

```
        Else if firstChar equals "r"
            irs.rtrv(SSN)

    Close file
    End clock
    Calculate duration
    Print stats


Hashtable:

Template HashTable
    init int tableSize
    SLL<V> array w table size

    Constructor HashTable()
        Set tableSize to 3 and make SLL w/ tablesize

    Constructor HashTable(size)
        Set tableSize to size and make SLL w/ tablesize

    bool find(item: V)
        init int index = hashfunc(item)
        Return result of table[index].search(item)

    bool insert(item1: V, item2: V)
        init int index = hashfunc(item1)
        If find(item1) returns true
            Return false
        Else
            table[index].insert(item1, item2)
            Return true

    bool erase(item: V)
        init int index = hashfunc(item)
        Return result of table[index].remove(item)

    int getSize()
        init int totalSize = 0
        For each index from 0 to tableSize - 1
            +1 totalSize by table[index].getSize() (if there is a node at the ind)
        Return totalSize

    int hashfunc(item: V)
```

```
        init int hash = 0
        For each char c in item
            hash using (hash * 31 + c) % tableSize
        Return hash
```

```
SLL
class SLL
    Pointer to Node<U> headPtr
    Init int size

    Constructor SLL()
        Set headPtr to nullptr
        Set size to 0

    Destructor ~SLL()
        Set current to headPtr
        While current is not nullptr
            Set next to current->next
            Delete current
            Set current to next
        Set headPtr to nullptr

    func insert(item1: U, item2: U)
        Create new Node<U>
        Set newNode->SSN to item1
        Set newNode->name to item2
        Set newNode->next to nullptr

        If headPtr nullptr
            Set headPtr to newNode
        Else
            Set temp to headPtr
            While temp->next is not nullptr
                Set temp to temp->next
            Set temp->next to newNode
        Increment size by 1

    Pointer to Node<U> search(item1: U)
        Set temp to headPtr
        While temp is not nullptr
            If temp->SSN equals item1
```

```
            Return temp
        Set temp to temp->next
    Return nullptr

bool remove(item1: U)
    Set current to headPtr
    Set prev to nullptr
    While current is not nullptr
        If current->SSN equals item1
            If prev is nullptr
                Set headPtr to current->next
            Else
                Set prev->next to current->next
            Delete current
            size -1
            Return true
        Set prev to current
        Set current to current->next
    Return false

int getSize()
    Return size

func display()
    Set temp to headPtr
    While temp is not nullptr
        Output temp->SSN
        Set temp to temp->next
```