BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Work Integrated Learning Programmes Division

Data Structures and Algorithms

Lab Sheet -3 Topic: Hash Table (PAN record holder)

General Instructions for Programming :

1. All inputs to the program must be through input txt files.

2. Use standard Java coding conventions. Each class: Node, Hash Table and Main code should be written in separate .java files.

Problem Statement:

Permanent Account Number (PAN) is a ten-digit alphanumeric number, issued in the form of a laminated card, by the Income Tax Department. PAN is an all India, unique number permanent for an individual's life, and will not change.

The PAN has the following format:
AAAAA9999A'A' in this format implies an upper case alphabet and 9 implies a numeral (digits).
The fourth character of the PAN must be one of the following, depending on the type of assessee:
C — Company
P — Person
H — HUF(Hindu Undivided Family)
F — Firm
A — Association of Persons (AOP)
T — AOP (Trust)
B — Body of Individuals (BOI)
L — Local Authority
J — Artificial Juridical Person
G — Govt

Assume the following details are stored for each PAN.

1. PAN number: 10 letter PAN (a string)
2. Name: Name of the person or Name of the organization (not exceeding 30 characters)
3. Place: Place where person stays or place where organization is registered. (not exceeding 50 characters)

The details of people holding PAN numbers is stored as a list in a HashTable for efficient recovery. The hashing mechanism used here is a two level hashing, where the first level determines the fourth character of the PAN concerned and hashes the corresponding record to the corresponding hash table. Fourth character of the PAN can be one of {C, P, H, F, A, T, B, L, J, G}. At the second level the record corresponding to the PAN is hashed in linear array (of the relevant records). We use quadratic probing to resolve the collision occurring here. The hash table is extended (rehashing all the elements into a new hash table) by an appropriate size, when an element cannot be hashed into a hash table.

Data Structures

1. Account: Contains the details of the PAN card holder (PAN number, Name and Place) as described earlier.

2. FL_HashTable: Each Location in this hash table refers to a separate SL_Hashtable.
3. SL_HashTable: There is one SL_HashTable Corresponding to each of 10 categories. Each SL_HashTable consists of those Account structures hashed into it.
4. Hash_Size: An array of 10, containing the size of each of the SL_HashTable's

   Size. All entries initialized to 13.

Functions:
1. void initializeHash(): Initializes FL_ HashTable and SL_HashTables with empty values.
2. int hash(String pan): Returns the hash value (ascii folding) for the corresponding hash table depending on the type of PAN number.
3. void addEntry(Account a): Adds an entry to the hash table. This function adds Account a into an appropriate SL_Hashtable. In case of collision, quadratic probing is used to resolve the collision. If quadratic probing fails to find a position in the SL_Hashtable, it calls extend_rehash() for this hash table and resizes the hash table before attempting to insert again.
4. void extend_rehash(int i) : extends the 'i'th SL_Hashing Table by appropriately doubling its size and rehashing the older contents into the extended hash table. This Extended SL_HashTable replaces the old 'i'th SL_HashTable.
5. void searchDetails(String pan): searches an entry 'pan' to the corresponding hash table depending on the type of PAN number and outputs the search result to an output file. The input for the search can be read from a file which contains a list of PAN numbers (one per line) that needs to be searched.

Input:
1. The input to the program (to be read from an input file) has the following format:
   Each line of the input file consists of 3 columns – first column being PAN, second column Name, and third column Place. This basically is the data that you need to put into the hash table.

2. Search.txt : containing the list of PAN numbers that need to be searched. Each PAN number that needs to be searched will be in a separate line in search.txt

Output:
For each input line in the file "search.txt" output a line which says whether the PAN
was found or not.
For example, if the PAN "BLJPT7673W" is found then you output a line which says
"The entry BLJPT7673W does exist – Arun Bangalore"
otherwise you should output a line which says
"The entry BLJGT7673W doesn't exist"

Deliverables:
1. Account.java – containing the node for PAN details and basic functions
2. Hash.java – containing the hash table definitions and corresponding operations
3. PANMain.java – containing the public static void main function, operation menu and switch cases with input requests where relevant.

Required Text Book Reading:
1. Section 2.5 of Algorithm Design by Michael T. Goodrich and Roberto Tamassia