BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Second Semester 2014-15

CS F211 Data Structures and Algorithms

Lab Sheet – 6

General Instructions for Programming

1. All inputs to the program must be either (a) command line arguments (b) or read from a file (other than stdin). DO NOT READ anything from stdin and DO NOT USE PROMPTS like "Please enter a number …".

2. You are required to write the output to a file (other than stdout) and errors if any to a different output channel (stderr or another file).

3. Use standard C coding conventions for multi-file programs. Separate the following: interfaces of functions (use a ".h" file), data type definitions (use another ".h" file), algorithm implementation (use a ".c" file), and driver/test code (use another ".c" code). In general, each module has to be written in **separate** c files.

4. All files related to a lab **must** be put inside a single directory by the name of the lab (lab1, lab2, etc.).

5. Valid makefile must be present in the directory.

6. Ensure that all the code written by you are compiling correctly. Preferably use gcc with the options **`-W -Wall -O2`**, while compiling your code.

Problem

A retail chain keeps track of customer's purchase pattern for marketing purposes. The chain maintains details such as customer's ID, item purchased and cost of the item purchased, to enable their research.

Write a program that can store the tuple (customer_id, item_code, item_cost) for fast searching by using hashing scheme mentioned below. Item_code should be a string of 8 characters.

Hashing Scheme:

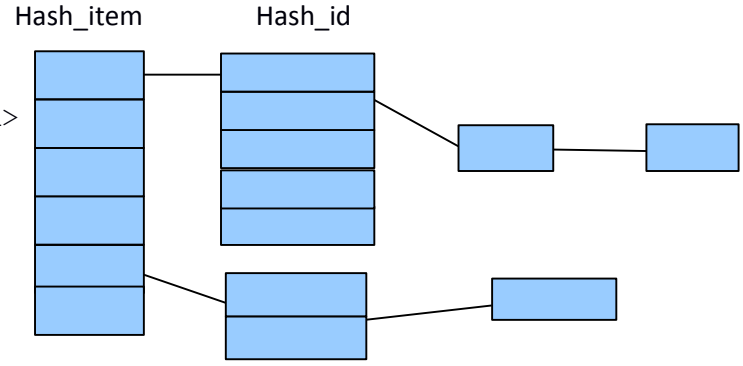
All data is hashed based on the item_code onto the hash table, HT_item, maintained using hash value of item_code. The entries in this table are further hashed on the basis of customer_id and stored in the hash table, HT_id, on the basis of the customer's id.

The input file format is as follows:

```
<cust_id> <item_code> <cost_item>
```

For ex: *inputHashing.txt* will look as follows

```
1028 huefena 89
9182 xyqiejp 34
1028 mnwivfd 92
```



You are required to perform certain queries based on the data provided in the input file.

Data structures to be used: (To be written in hashtable.h)

- Hash_item: A dynamic array to store the hash table based on item code. This table internally keeps track of the second hash table, Hash_id, for hashing customer's id. The item_code is hashed using the function: ((sum of ASCII values mod p) mod SIZE_T). Here, SIZE_T refers to the size of the table and p is a prime number larger than SIZE_T.
- Hash_id: A hash table of fixed size, CSIZE, that is used to hash entries based on customer's id. Each entry in this table contains the list of the tuples to be inserted for the specific cust_id and item_code and the cost of the item. The hash function used is: sum_of_digits(cust_id) mod CSIZE
- Node: This is used to maintain the list of the tuple(cust_id, item_code, item_cost) in in Hash_id.

Problem Definition:

1. Read all data from input file and store them on to the hash tables based on the item_code and customer's id.
2. Enable simple operations on the hash table like insert and find.

Operations: (To be written in header file htops.h and implementation in htops.c)

a. Hash_item * populateHashtable(char * filename): This function reads the contents of the file, creates the hash table and populates the hash table with the tuples from the input file. This function returns the newly created hash table.

b. void printHT(Hash_item ht, int htsize, FILE *fp): This function prints the contents of the two hash tables, and their entries into the fie given by fp. [Note: You may pass stdout as the last parameter for debugging purposes.].

c. insertHashtable(Hash_item ht, int htsize, int cust_id, char *item_code, int cost_item): This function creates a node for the tuple (cust_id, item_code, cost_item), and makes suitable entries into the hash tables. This function is used by fn (a) to populate the hash tables.

d. int findEntry(Hash_item ht, int htsize, int cust_id, char *item_code): This function searches for an entry corresponding to customer's id, cust_id, and item_code. This function returns the cost of the item purchased by the customer if found; else it returns -1.

e. int purchasedby(Hash_item ht, int htsize, int cust_id): This function traverses through the two hash tables and computes the total cost of the items purchased by the customer, cust_id.

f. int hash1(char *key, int size): This function computes the hash value for use in Hash_item.

g. int hash2(int key, int size): This function computes the hash value for use in Hash_id.

Tasks to perform:

(a) Read a given number of records from input file (*inputHashing.txt)*.

(b) Write the header files hashtable.h and htops.h

(c) Write the functions mentioned earlier in the file htops.c

(d) Write a driver file named query.c that performs the following:

i. Read the input filename, the output filename, and the customer's id and item code for the two queries, as command-line parameters.

ii. Read the input file and populate the hash table.

iii. Perform the first query of finding the total amount purchased by a specific customer, and store the result in the output file.

Deliverables:

hashtable.h, htops.h, htops.h, query.c, and the input files that you have used for testing.

Take home assignment:

1. Write a function to query the total value purchased by a specific customer.
2. Write functions for different probing techniques for second hash table (Hash_id).