CSE-310 : Data Structures and Algorithms

Homework 9 - Due Tuesday May 3rd by 1:30pm

This is an individual programming assignment. The goal is to implement hash tables with insert and search capability using chaining and open-addressing for collision resolution. The input file will contain distinct key values that should be inserted into four separate hash tables followed by a key value that is searched for in all tables. A separate output file should be created for every hash table. Your program should be compilable on the general.asu.edu server and must include a README file describing how to compile and test your program.

- Your program will be tested on the general.asu.edu server so make sure that it compiles and executes correctly on that server.
- Your program must accept the name of the input file as an input parameter.
- The output files should be named "chain.txt", "linear.txt", "quadratic.txt", and "double.txt".
- You *must* include a documentation file named README that includes your name and student ID and explains how to compile and test your program. You can use a "makefile" to simplify the compilation process.

1. Input file format
The Input file will only contain 32 bit-integer values. The first portion of the input file will contain the list of keys that should be inserted into the hash tables which will be preceded by a positive number which is the *number of key values* that should be read for insertion. You can assume key values are distinct positive integers. The second part is a single key value that should be searched for inside each of the tables.

2. Hash tables
Four hash tables should be implemented. Following is the parameters that should be used for each of these tables.

2.1 Chaining
The first hash table will use chaining for collision resolution. Use the hash function $h(k) = k \ mod \ m$ where $m = 23$.

2.2 Linear
The second hash table will use open-addressing for collision resolution with linear probing sequence. Use auxiliary hash function $h'(k) = \lfloor m(kA \ mod \ 1) \rfloor$ with $A = 0.618$ and $m = 32$ and hash function $h(k, i) = (h'(k) + i) \ mod \ m$ where $m = 32$.

2.3 Quadratic

The third hash table will use open-addressing for collision resolution with quadratic probing sequence. Use auxiliary hash function $h'(k) = \lfloor m(kA \ mod \ 1) \rfloor$ with $A = 0.618$ and $m = 32$ and hash function $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \ mod \ m$ where $m = 32$ and $c_1 = c_2 = 0.5$.

2.4 Double Hashing

The final hash table will use open-addressing for collision resolution with double hashing probing sequence. Use auxiliary hash functions $h_1(k) = k \ mod \ m$ where $m = 31$ and $h_2(k) = 1 + (k \ mod \ m')$ where $m' = 30$ and hash function $h(k, i) = (h_1(k) + i h_2(k)) \ mod \ m$ where $m = 31$.

3. Output file format

A separate output file should be created for each table. Each file will contain the final state of the hash table *followed by a* -1 *delimiter* and then the information regarding the search process. You can use delimiters to make the output more readable but don't add any extra numbers in your output file.

3.1 Chaining

The file should be named "chain.txt" and contain the index of each cell (starting from zero) followed by the key values inside the link list of that cell starting from the head of the list. Print a -1 delimiter after printing the current state of the entire table. When searching for the final key value print the index of the cell which is being pointed to by the hash function followed by the key values which are encountered inside the link list while performing the search. Print all of these key values (including the search key if found) with a space delimiter.

3.2 Open addressing

These files should be named "linear.txt", "quadratic.txt", and "double.txt" which correspond to the probing sequence. In each of these files the current state of the hash tables should be saved as the index of each cell (starting form zero) followed by the key value if the cell is occupied or -1 if it's empty. Add a -1 delimiter after printing the entire table. When searching for the final key value print the *index of each cell* which is looked at while searching for the given key. If the search ends without finding the search key add a -1 to the end of this sequence.

4. Submission

Create a zip file named "lastname_firstname_hw9.zip" that consists of your program files and the README file and upload it to the hw9 section in the assignments page before the deadline. Turn in a hard-copy of your README file in class.