Data Structure and Algorithms *CSE310-Yang, Fall 2017*

# HW 03: Programming assignment-Hashing

**Electronic copy** *due at end of day on Monday, Oct 16, 2017 on Blackboard*

This is a programming assignment. You are asked to implement some of the operations related to hashing. You are only allowed to use C++ as your programming language. Although you may develop your code on your personal computer, you have to make sure that it compiles and runs correctly on **general.asu.edu**. You need to submit this assignment electronically through Blackboard. We will grade your assignment using the electronic copy you submitted.

In all of the procedures below, assume that the keys to be stored in your hash table (by chaining in the former item, or by open addressing in the latter two items) are natural numbers in the range $[0, 100]$. You need to implement the following procedures, whose description and pseudocode were given in class:

- Chained-Hash-Insert and Chained-Hash-Search, using the hash function $h(k) = k \bmod p$, and chaining for resolving conflicts on a hash table of size $p$.
- Hash-Search-Linear and Hash-Insert-Linear, using open addressing for resolving conflicts on a hash table of size $p$, and linear probing with $h'(k) = \lfloor p((\alpha * k) \bmod 1) \rfloor$ where $\alpha$ is a real number. (therefore the probe sequence will be defined by $h(k, i) = (h'(k) + i) \bmod p$, for all $0 \leq i < p$).
- Hash-Search-Double and Hash-Insert-Double, using open addressing for resolving conflicts on a hash table of size $p$, and double hashing with $h_1(k) = k \bmod p$ and $h_2(k) = 1 + (k \bmod (p-2))$ (the probe sequence will then be defined by $h(k, i) = (h_1(k) + i * h_2(k)) \bmod p$, for all $0 \leq i < p$).

In each of the items below, you will insert the sequence of keys in the order that they appear in the input file into an initially empty hash table and store it in a separate output file.

1. Chained-Hash-Insert. Print out the linked lists headed by each slot of the table (one linked list per line, with space between consecutive elements of the list), in increasing order of the indices of the slots, right after you have finished inserting the sequence of keys. Indicate at the beginning of each list to which slot it corresponds to. Now search for the key $k_1$ (read from the input file) in your table and print out the index of the slot that heads the linked list where this key was found.
2. Hash-Insert-Linear. Print out the contents of each slot in your hash table in increasing order of the indices of the slots, right after you have finished inserting the sequence of keys. Now search for the key $k_1$ (read from the input file) in your table and print out the sequence of slots probed in the search until you find this key.
3. Hash-Insert-Double. Print out the contents of each slot in your hash table in increasing order of the indices of the slots, right after you have finished inserting the sequence of keys. Now search for key $k_1$ (read from the input file) in your table and print out the sequence of slots probed in the search until you find this key.

Following is the format of the input and output files you should use for your program. One way of checking whether your program is working correctly is to test it on the sample input file on the blackboard and compare your output files with the corresponding output files on the blackboard.

**Your program should be well documented. At the start of each function, you should inform the reader what this function is about and what algorithm(s) you are using.**

**Format of the Input File:** This file will be given as a parameter to the program. There will be a single value in each line of the input file. The first line will contain the value for $p$ and the second line will contain the value for $\alpha$ (the real number used in linear hashing). The next line will be the number of keys that need to be inserted into the hash table which we will call $n$. The subsequent $n$ lines will contain the actual keys. Finally the last line will contain the value for $k_1$ which is the key that your program should be searching for after it has inserted all of the $n$ elements into the hash table.

**Format of the Output Files:**

- *Chained Hash:* The first $p$ lines of this output file will represent the $p$ slots of the hash table. In the beginning of each line will be the index of that slot (0 through $p-1$) followed by a ":". Then for each slot, output the elements according to the order they appear on the linked list. (New elements will be inserted at the head of the linked list.)

  Next, output the slot that contains key $k_1$.

- *Linear Hash:* The first $p$ lines of this output file will represent the $p$ slots of the hash table. In the beginning of each line will be the index of that slot (0 through $p-1$) followed by a ":". Then for each slot, print out the element in that slot, if there exists one.

  Next, print the *sequence of probes* made for key $k_1$ in a single line with space delimiters.

- *Double Hash* The first $p$ lines of this output file will represent the $p$ slots of the hash table. In the beginning of each line will be the index of that slot (0 through $p-1$) followed by a ":". Then for each slot, print out the element in that slot if there exists one.

  Next, print the *sequence of probes* made for key $k_1$ in a single line with space delimiter.

You should also prepare a Makefile which will compile your program and make sure your executable file is named "Hashtable".

**Grading policies:**
If you program does not compile on general.asu.edu, you will be deducted at least 50 pts on this project.

- (5 pts) IO operation.
- (5 pts) Makefile and documentation, also make sure your program is named "Hashtable", otherwise more points will be deducted.
- (30 pts) Chained Hash.
- (30 pts) Linear Hash.
- (30 pts) Double Hash.

You will need an ssh client to connect to the general.asu.edu linux server and to upload your program. You can either write the program on another computer and upload it or use the vi editor on the server to write your program.