Data Structures
Programming Homework 5

Won Kim

(Lecture by Youngmin Oh)

Spring 2022

Programming Homework 5

- Static Hashing (30 points)
- Internal Sorting (30 points)

Problem 1: Hashing (30 points)

- Write a hash table management program in C.
- The program inserts keys into a hash table, deletes a key, and searches a specific key in the hash table.
- You will use overflow chaining for key collisions.
- The hash table is a struct array of 33 buckets, with no overflow buckets.
- Create the hash table with the keys given in the **keyinput.txt** file.
- To hash, first sum the ASCII code of each alphabet in the key, and then use the modulo function (sum % 33).

Problem 1: (cont'd)

- After creating the hash table, do the following (and print each result):
  - Print the contents of the hash table (including the overflow) for each index (If the bucket is empty, print "NULL")
  - Search for Blue, black, Purple (print "found" or "not found")
  - Delete Purple, Blue, Green (print "deleted" or "not found")
  - Insert Green, White, Golden, nedloG (print "exists" or "inserted")
  - Search for Blue, nedloG, Yellow, Green (print "found" or "not found")
  - Print the contents of the final hash table (including the overflow) for each index (If the bucket is empty, print "NULL")

Notes

- (10 points) First, design the overall structure of the program
  - Draw example scenarios for search, insert, and delete.
  - Specify each of the functions needed, with maximum reusability of the most basic functions.
- (20 points) Second, implement (code) and run the program.
  - Be sure to comment all the functions, and observe the coding guidelines (layout, naming, indentation, etc.).

Problem 2: Sorting (30 points)

- Part 1: Create 2 tables
  - Create 2 tables as struct arrays
    - Table1 (with 15 rows and 3 columns, each integer type)
    - Table2 (with 10 rows and 3 columns, each integer type)
  - Fill the tables with random integers between 1 and 100.
  - Print the 2 tables (in a nice table form)
  - Update the tables
    - Table 1, row 10 & 14 column 3 (change the value to 55)
    - Table 2, row 5, column 1 (change the value to 55)

Problem #2: (cont'd)

- Using the Table 1 and Table 2 prepared in Part 1,
- Part 2-1: sorting
  - Write a duplex selection sort program
  - Using the sort program, sort Table 1 (on column 3) and Table2 (on column 1)
  - Print the 2 tables
- Part 2-2: sorting
  - Use quick sort (you may search the Internet for a good source code)
  - Sort the 2 tables (as in Part 2-1) and print them.

Problem #2: (cont'd)

- Part 3: merge join of the two tables from Part 2
  - Write join (t1c, t2c, val), a program that joins 2 tables,
    where t1c is the array index for a column of table1,
    and t2c is the array index for a column of table2;
    and val1 is the user-specified value for the t1c column
  - The join function concatenates the row of table1 with a row of table2, when the t1c column value and the t2c column match.
  - ** A naïve way to join 2 tables requires15 x10 comparisons of the column 3 values of Table 1 with the column 1 values of Table 2.
  - ** Because the two tables are sorted, you can end the search in both tables early.

Table 1

c1	c2	c3
		54
		55
		55
		56
		56
		57

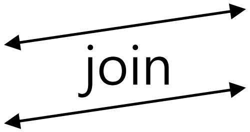

Table 2

c1	c2	c3
54		
55	??	??
56		
57		

Problem #2: (cont’d)

- Part 3: merge join of the two tables from Part 2
    - Test that the join function joins the 2 tables based on the value 55 for column 3 of table1 and the value of column1 of table 2. (Print the concatenated rows.)

Notes

- (10 points) First, design the overall structure of the program
  - Specify each of the functions needed, with maximum reusability of the most basic functions.
  - State key algorithm design points.
- (20 points) Second, implement (code) and run the program.
  - Be sure to comment all the functions, and observe the coding guidelines (layout, naming, indentation, etc.).