CS 2124 Data Structures

Programming Project 5

In this project, we will be storing and updating the win/loss records of the NBA teams from the 2018-2019 season using a Hash Table. We will store/retrieve the team’s information using their team name as the key value that will be converted into a valid index in our array.

**Files provided in the project:**

1. **HashTable.h**. This file contains the prototypes and definitions for the functions for a hash table. You need to implement the following functions in the file HashTable.c:
   1. **HashTable newHashTable(int n).** Malloc a new HashTableImp, malloc the hashTable to be an array of Elements of size n, initialize each chainIndex to be -2 (indicating that the spot is empty), and return a pointer to the HashTableImp.
   2. **void freeHashTable(HashTable h).** Free the hashTable pointer and then h itself.
   3. **int stringToInt(char \*stringToConvert).** Given a string, convert it into an integer to be used in either the division method or the midsquare method.
   4. **int divisionMethod(int key, int n).** Given a key value and the size of the hash table, use the division method to find a valid index for hashTable.
   5. **int midsquareMethod(int key, int n).** Given a key value and the size of the hash table, use the midsquare method to find a valid index for hashTable.
   6. **void put(HashTable h, Element e)**. Insert e into our HashTable h by using stringToInt to convert the team name into an integer and then passing that integer to one of the division or midsquare method functions. If this entry is not occupied, insert it there and change the corresponding chainIndex to be -1. If there is a collision, use open chaining to find an open location for e, and update the chainIndex values accordingly.
   7. **int get(HashTable h, char \*teamName, Element \*e)**. Find the entry in the HashTable h that contains the data for teamName. Return the struct for this team through e which is an Element that has been passed by reference. Functionally return TRUE if the team exists and return FALSE if it does not exist.
2. **HashTableEC**.**h** This is for Extra Credit. If you are not doing extra credit then you can ignore this file. For more on extra credit, see below.
3. **nbaTeamNames.csv.** This file contains each team from the 2018-2019 NBA season. Each team is listed one per line.
4. **nbaData2019.csv**. This file contains the results for each game from the 2018-2019 NBA regular season. It is a comma separated file, so your computer may want to use Excel (or some other spreadsheet software) to view the file. Open it in a text editor to see the format that your program will see. Each line contains the results of one game in the following format: “**Team1,Score1,Team2,Score2**”. Note that the team names will have spaces in them, so you use the position of the commas to tell when the team name has ended and the score has began. Score 1 was the final score for Team 1, and Score 2 was the final score for Team 2.
5. **abc123Project5Part2.c**. Rename this file to your abc123. Inside main, you should first create a hash table of size 100 (there are 30 teams – making the hash table sufficiently large reduces the chances of collisions). Then open nbaTeamNames.csv and read in each of the teams, and for each team, use the “put” function to write an Element for the team with an initial win/loss record of 0 wins and 0 loses. Then open the nbaData2019 file. Read in one line, determine the teams that played in the game and their corresponding scores, determine which team won the game and which team lost (high score wins). Then use the “get” function to obtain the Element associated with the team, and increment the win total if the team won, and increment the loss counter if the team lost. After all games have been considered, reopen the nbaTeamNames.csv file (or just store them in an array initially) to print out the Team name and the Win and Loss counts, one team per line. Note that you should use the “get” function with the team names here to do the printing rather than using a for loop to iterate through each index of the hashTable. Print the results in the following format: “San Antonio Spurs: XX Wins – XX Losses”.
6. **Makefile**. Update the makefile to reflect your abc123. Compile using **make**. Execute the program using **./p5**

**Extra Credit:**

Implement a hash table that handles collisions by maintaining a linked list of all of the synonyms of a given index (e.g., myHashTable -> hashTable[36] contains the head pointer of a linked list that stores every value that hashed to index 36). We have provided the file HashTableEC (for HashTable extra credit). Create a file HashTableEC.c to implement the functions, and create a new file abc123Project5EC.c to implement the same project as above only using your HashTableEC data structure (this main file can be almost identical to the main project file).

**Submitting:**

Zip up your project folder in a file name abc123Project5.zip (with your own abc123) and submit on the dropbox on Blackboard by the due date.