ECE 361 Fall 2019
Homework #3

This assignment is done individually and is worth 100 points (+ up to 3 pts extra credit for implementing additional useful functionality in Question 2).

THIS ASSIGNMENT SHOULD BE COMPLETED BY 10:00 PM ON SUN 10-NOV. WE ARE USING GITHUB CLASSROOM FOR THIS ASSIGNMENT SO MAKE YOUR FINAL COMMIT TO YOUR GITHUB PRIVATE REPOSITORY FOR THE ASSIGNMENT BEFORE THE DEADLINE. ALSO, PLEASE UPLOAD A .ZIP FILE OF YOUR FINAL REPOSITORY TO THE HOMEWORK #3 DROPBOX IN D2L. C SOURCE CODE FOR YOUR PROGRAMMING SOLUTIONS SHOULD BE TEXT FILES (NOT .DOCX, ETC.) THAT HAVE A `.c` EXTENSION. HEADER FILES SHOULD BE TEXT FILES THAT HAVE A `.h` EXTENSION. YOUR TRANSCRIPTS (LOGS) SHOULD BE SUBMITTED AS TEXT FILES (`.log`) BY EITHER REDIRECTING THE OUTPUT FROM YOUR SHELL TO A FILE OR BY USING THE BASH TEE COMMAND TO SEND OUTPUT TO BOTH THE CONSOLE AND A FILE (EX: $ echo "hello world" |tee helloworld.log). QUESTIONS THAT DO NOT REQUIRE SUBMITTING CODE MAY BE WRITTEN IN YOUR FAVORITE WORD PROCESSOR AND SAVED/SUBMITTED AS A .PDF FILE. NAME ALL OF THE FILES IN THE REPOSITORY WITH DESCRIPTIVE NAMES. BE SURE YOUR CODE IS ORGANIZED INDENTED APPROPRIATELY, USES MEANINGFUL VARIABLE NAMES, AND INCLUDES COMMENTS THAT AID IN UNDERSTANDING YOUR CODE.

Question 1 (20 pts): Hashing and Hash Tables

Answer the following questions about hash tables.

a. (5 pts) Describe a Hash table and state what is used for. List several advantages and disadvantages of using a hash table.

b. (5 pts) Circle all of the correct answers, -1 pt for each incorrect answer) Given the following input `(4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199)` and the hash function `x mod 10`, which of the following statements are true?
   - 9679, 1989, 4199 hash to the same value
   - 1471, 6171 hash to the same value
   - 4322, 1334, 6173 hash to the same value
   - All elements hash to the same value
   - Each element hashes to a different value

c. (5 pts) Briefly describe the difference between conflict resolution by Chaining and conflict resolution by Open Addressing. List a few advantages and disadvantages of each.

d. (5 pts) What are the average and worst case `O(n)` for:
   - Inserting items into a hash table?
   - Deleting items from a hash table?
   - Looking up key in a hash table?

Question 2 (60 pts): Using Abstract Data Types

This question gives you an opportunity to make use of the Linked List ADT you developed in Homework #2 into practice. Our target application builds and operates on a database of professional sports team that you will store and operate on in a Linked List. Since you attend school in Portland and Portland is fondly known as "soccer city USA"

https://www.nbcsports.com/northwest/portland-timbers/top-5-reasons-portland-indeed-soccer-city-usa

2019 MLS regular season attendance: https://soccerstadiumdigest.com/2019-mls-attendance/

2019 NWSL regular season attendance: https://soccerstadiumdigest.com/2019-nwsl-attendance/

our database contains information about professional soccer teams in the MLS. Information for the database will come from a text file that your program will read and parse. Each record in the file has the following fields:

- A `char[] array` containing the name of the team. Underscores are used to join individual words (ex: Portland_Timbers) so the full name can be scanned as a single string (%s). The underscores should be replaced by spaces when you display the record.
- A `char[] array` containing the nickname (abbreviated name) for the team. Can be scanned as a single string (%s)
- An `int` encoding the league the team plays in (OTHER, MLS, NWSL, USL)
- An `int` encoding the conference the team plays in (Other, Eastern, Western, NWSL)
- Three `ints` (`int-int-int`) encoding the win-lose-draw record for the 2019 regular season

Lines in the file that start with `//` are comments and should be ignored. Blank lines should also be ignored.

a. (0 pts) Study the starter code provided with the project. It is important to understand the architecture of the application and to identify what functions you need to edit or create to create a working application.

b. (15 pts) Rewrite your Linked List ADT from Homework #2 to incorporate the team info record as `data` for the linked list instead of an `int` as in Homework #2. Modify the header (`.h`) file for the Linked List ADT that is included in the project release for your Linked List ADT. You may want to add functions to your API or modify the definition of the function to better meet the needs of your application. It is OK to do this for HW #3 as long as your code is organized and documented so that the instructor and/or T/A can better understand your implementation. *NOTE: I have included my Linked List ADT from my Homework #2 solution in the project release but my preference is for you to rewrite/exercise the Linked List ADT that you wrote.*

c. (15 pts) Write the code for the missing functions listed in the two header files. A function that reads the team info text file is included in the project release but there are several functions for you to write/debug.

d. (15 pts) Write a `main()` function that populates your Linked List(s) with the team info records from the file `data_file/teaminfo.txt`. At a minimum, `main()` should implement the functionality listed below. You may provide additional functionality (up to 3 pts of extra credit on the assignment):

- Display all of the teams by conference. For example:

  MLS (Eastern Conference)
  New York Red Bulls
  D. C. United
  Toronto FC
  …

  MLS (Western Conference)
  Portland Timbers
  Vancouver Whitecaps FC
  Seattle Sounders
  …

  Hint: You may want to instantiate/populate two Linked Lists, one for the Eastern conference teams and one for the Western conference teams to simplify this task.

- Find and display the 5 teams with the best winning percentages in 2019 [wins / (wins + losses + draws)] no matter which conference they are in.

- Display the team with the most wins and the team with the most losses in each conference.

e. (5 pts) Write a `make file` that you can use to compile and build your application.

f. (10 pts) Build your application using the `make file` you wrote in part c. Execute your program. Include a transcript (log) that shows your program being compiled and executed. The transcript should demonstrate that your program is working correctly.

Push all of your source code (`.c` and `.h`) files and transcripts to your GitHub repository for the assignment.

LOOKING AHEAD: ORGANIZE, SAVE, AND DOCUMENT YOUR WORK. WE WILL LIKELY REVISIT THIS APPLICATION IN HOMEWORK #4.

Question 3 (20 pts): `make` and `makefile`

Answer the following questions about the `make` utility and make files. *Note: these questions were taken from an online list of software engineering interview questions.*

a. (4 pts) How is the `make` utility used in the software development process? List several ways that `make` can be used to simplify the compile/build cycle for software projects containing multiple modules.

b. (4 pts) When a target of `makefile` fails to execute:
   a) `make` does not execute any other target dependencies on it
   b) `make` returns a status
   c) both a) and b)
   d) none of the above

c. (4 pts) The command `make google` will:
   a) Create the executable if `google.c` is present in the current directory
   b) Create the object file named `google.o`
   c) Give an error
   d) none of the above

d. (4 pts) The `makefile` starts executing from the:
   a) First target
   b) First target whose name starts with "." (a period)
   c) First target whose name does not start with "." (a period)
   d) none of the above

e. (4 pts) When we type `make` on a terminal:
   a) `make` reads the file named `makefile` in the current directory
   b) `make` reads the file name `Makefile` in the current directory
   c) `make` reads the file `makefile` in the parent directory
   d) a) and b)
   e) a) and b) and c)

<finis>