Data Structures and Algorithms, CS146, Spring 2020
Programming Assignment III
Micro Version of Facebook

Due at 11:59pm, on May 12th (Tuesday), 2020
(No late submission! 0 for late submissions and email submissions!)

Warning:
This programming **assignment is for individual work only. You are not allowed to use someone's code** or copy from internet. Code plagiarism checker tools (Jplag, MOSS, etc.) will be used to check the similarity of codes. **You will be asked to demo and explain your code to the instructor or to the class. Cheating will not be tolerated** and will be reported to University. Please check on the University Policies in the syllabus. This document is under copyright protected. You are not allowed to upload this document to any websites, otherwise, cheating will be treated (report to SJSU) and your course grade will be changed to F, even after you graduated.

Problem Statements

Facebook is a very popular web based social media application. In this programming assignment, you have to use **Hash Table and Hashing Functions** to design and implement a micro-version of Facebook.

Specifically, your program must provide options (or GUI) for users to choose the following operations: (**Item 1-5 must implement Hashing**)

1. Create a Person record/class containing Name and a linked list of Friends. (You may assume that no two people have the same name.)
2. Record a person as a new friend. (Make a friend.)
3. Remove a person from the friend list. (Unfriend)
4. Search a person's name to list his/her friend list. (he/she may or may not be your friend.)
5. Enter two person's names to check whether the two people are friends. (Print "Yes" or "No")
6. Using Binary Search Tree (BST) to sort and list a person's friends in alphabetical order.

Functional Requirements

1. For this programming assignment, your job is to design and implement a micro version of Facebook using Hash Table. The Hash table MUST contain **minimum 10 slots and maximum 15 slots. At least 50 distinct users/friends must be stored in the table by using a hash function. Other size of table and users will be considered violating functional requirements and 0 point will be given.**

2. The following 50 distinct popular names MUST be used to test your program (-20% if using other names.). You may do your own way to make friendships for these people. (Remember that you have to convert a name into a natural number for hashing purpose)

Rank	Male name	Female name
1	Liam	Emma
2	Noah	Olivia
3	William	Ava
4	James	Isabella
5	Logan	Sophia
6	Benjamin	Mia
7	Mason	Charlotte
8	Elijah	Amelia
9	Oliver	Evelyn
10	Jacob	Abigail
11	Lucas	Harper
12	Michael	Emily
13	Alexander	Elizabeth
14	Ethan	Avery
15	Daniel	Sofia
16	Matthew	Ella
17	Aiden	Madison
18	Henry	Scarlett
19	Joseph	Victoria
20	Jackson	Aria
21	Samuel	Grace
22	Sebastian	Chloe
23	David	Camila
24	Carter	Penelope
25	Wyatt	Riley

3. **Using Hashing with Chaining** (page 258 in textbook):
3.1 Design and implement your own **Division Method** Hash Function: (Must show and screenshot the content/result of hash table for each of the following functional requirements)
    a) Define a "Person" class which has one field for the name and another field for the linked list of friends. (You may assume that no two people have the same name.) (-20%)
    b) Store the friends of each person in a linked list. A linked list class must be created by yourself, not from the Java library. (-20% if missing)

c) Create a global hash table that indexes each Person object under the name and show the result of hash table. (-50%)
d) Record a person as a new friend. (Make a friend.) (-20% if missing)
    i. Chained-Hash-Insert(T,x)
e) Remove a person from the friend list. (Unfriend) (-20% if missing)
    i. Chained-Hash-Delete(T,x)
f) Search a person's name to list his/her friend list. (he/she may or may not be your friend.) (-20% if missing)
    i. Chained-Hash-Search(T,k)
g) Enter two person's names to check whether the two people are friends. (Print "Yes" or "No") (-20% if missing)
h) Use BST (**must use pseudo code in textbook**) to sort and list a person's friends in alphabetical order (-30% if missing)

Programming Requirements

1. All functional requirements MUST be written in Java and it is required to use pseudo codes in the textbook. (Please be noted that you MUST use the pseudo codes provided in textbook to write your Java codes. Any other codes will be treated as failing requirements and **will receive 0 points**.)

2. Each java file/class/subroutine/function call **MUST** contain a header comment at the beginning of each and make sure you have enough comments at the end of lines in your code. (-20 points if your code did not provide enough comments.)

3. A professional report (clean and organized) is required in IT industry. You **MUST** write a **professional/formal** report in MS Word/PDF format with the following items included in the report: **(You will receive 0 point if a report is not submitted.)**

    a) Cover Page – PAIII Title and your name **(-10% if missing)**
    b) Explain Design and Implementation: (Ranging -20% ~ -40%)
    Briefly explain/illustrate how you design and implement your Hash Table and Hash Functions. such as diagram, graph, data structure, (Array, ArrayList, Linked List, Vector, etc.).
    c) A list of classes/subroutines/function calls: Explain purpose for each. **(-20% if missing)**
    d) **Self-testing screenshots and providing explanations: (-50% ~ -100 if missing)** Provide **enough screenshots of source code for each function including inputs and outputs** for each of exercise. You MUST explain your codes corresponding to the inputs and outputs of screenshots. This is to verify your codes and to check to see if your codes match the functional requirements and run correctly by yourself. (You

Copyright© Mike Wu

will receive 0 point if no self-testing screenshots provided. Points will be taken off if not enough self-testing screen shots provided.)

e) Show and explain your procedure (step by step) of how to unzip your files, install your application, and run/test your codes. (-20 points if not provided)

f) Problems encountered during the implementation: Describe all problems found/encountered during your implementation and how you resolved them. (-20% if not provided or too simple.)

g) Lessons Learned: Describe the concepts and skills you have learned from this programming assignment and any inspiration or motivation you got from this programming assignment. (-30% if not provided or too simple.)

Submission Requirements:

1. Zip all your files including your formal report, all *.java files, and a Java executable file (a jar file, so that I can run your code) into one zipped file with file name: **PA3_Last_Name.zip**, Any file with incorrect file name will be taken off 20%. **[No jar file or your code is not runnable (test it before submission: java -jar filename.jar): -40% if missing]**
2. The deadline to submit/upload your zip file to Canvas is before noon on 5/12/2020. After deadline, the submission link will be closed.
   Warning: DO NOT wait until the last minute to upload your files to Canvas. It has been happening all the time that it took forever to upload files to Canvas during the last hour/minute. **No excuse for late submission.**

Grading Criteria

Grading is based on the creative design, full functioning, completeness and clarity of codes and the **professional** report. Each grading policy is specifically listed after each functional item above. You should follow the functional requirements to avoid losing points.

Useful Reference Links:

1. Make Java executable Jar file

Note: Eclipse with JDK 8 and under can export project into Runnable Jar, but not JDK 9.

If you jar file is not runnable when I click on it, it is OK as long as it can be runnable when I use commend window and type "java -jar name.jar"