CSE 2010, HW1, Fall 2019

Due Thu Sep 5 at the start of your lab section; Submit Server: class = cse2010, assignment = hw1S*x*Individual

Due Thu Sep 5 at the end of your lab section; Submit Server: class = cse2010, assignment = hw1S*x*GroupHelp

x is 2 or 3—your section number (or j for java).

Consider a text messaging app that can send and receive messages to and from different contacts. Since we want to route a message from the sender to the recipient as soon as possible, two messages might use different routes according to the changing traffic conditions. As a result, messages might arrive at the recipient out of order according to their time stamps. However, we desire the app to display messages in the order of the time stamps, otherwise, a conversation might look incoherent.

For this assignment, assume the user has three contacts: Alice, Bob, and Carol. To separately manage messages for the different contacts, use different singly linked lists.

Input: To simulate the events for the text messaging app, an input file contains events in the same directory as your program file called hw1.c that has the main function. Your submission takes the input file name as a command-line argument. Each line is one of the following event:

- ReceiveMessage *timeStamp contact message*
- OpenApp
- DisplayConversation *contact*
- SendMessage *timeStamp contact newMessage*
- DeleteMessage *timeStamp contact*
- CloseApp

For simplicity, time is a unique non-negative integer in the format of HHMMSS. The user can open the app, display the conversation, send a message to a contact, delete a message from a contact, and close the app.

Output: The program prints events to the standard output (screen). Each event is on one line and possible events are:

- NotifyUser *contact*
- OpenApp
  Alice *numberOfNewMessages*
  Bob *numberOfNewMessages*
  Carol *numberOfNewMessages*
- DisplayConversation *contact*
  *timeStamp*1 me/*contact message*1
  *timeStamp*2 me/*contact message*2
  ...
- SendMessage *timeStamp contact newNessage*
  *timeStamp*1 me/*contact message*1
  *timeStamp*2 me/*contact message*2
  ...
  timeStamp me/*contact newMessage*
- DeleteMessage *timeStamp contact*
  *timeStamp*1 me/*contact message*1
  *timeStamp*2 me/*contact message*2
  ...
- CloseApp

When a message is received, the user is notified. When the app is opened, the number of new messages for each contact is displayed. After displaying the conversation of a contact, the messages are considered read/old. After sending or deleting a messsage, the updated conversation is displayed. For simplicity, no messages will be received while the app is opened. Sample input and output are on the course website.

Submission: Submit hw1.c that has the main function and other program files. Submissions from individual students are due at the beginning of their respective lab sections via assignment *hw1SxIndividual* (see the top). During the lab session on the due date, we encourage students to bring test cases (beyond the sample input) to test and improve each other's program in the group. Improved programs are submitted via assignment *hw1SxGroupHelp*, which is due at the end of the lab section (see the top). Your program is mainly evaluated based on *hw1SxIndividual*. Improvement on test cases will receive half credit. Specifically, $testCaseImprovement(hw1) = testCaseScore(hw1SxGroupHelp) - testCaseScore(hw1SxIndividual)$; $testCaseScore(hw1) = testCaseScore(hw1SxIndividual) + testCaseImprovement(hw1)/2$. Note the late penalty on the syllabus if you submit after the due date and time as specified at the top.