CSL 214 : Data Structures and Program Design II
H/W Assignment-1

1. Announcement Date: **23-03-2022**

2. Due date: Submit online by **11:59 PM on Wednesday 30th March, 2022**

3. The assignment has to be done individually.

4. For this assignment, Linked Lists are the preferred data structure. Hence, choose linked list (any type that you are comfortable with, singly linked list or circular linked lists or doubly linked lists) for any storage requirement you may have.

5. No Copying / sharing of code is allowed for this assignment. If any such case is identified, the original author and person who has copied, both will be penalised equally and zero marks will be awarded.

You need to submit your source files by attaching them to a mail and sending it on **dspd.assignment@gmail.com** by the common deadline. Please attach .c and/or .h and/or .txt files only. No .obj or.exe files should be attached. The subject should be marked as DSPD2-HW-Assignment-1: Your enrolment no.

Problem for R1 Batch (Enrolment Numbers BT20CSE001 to BT20CSE030):

The owner of the health club wants to store the records of the members, trainers and schedule the slots for them. As per the covid-norms set by the government and the floor capacity of the club, not more than 10 members can be present inside the club at any time. The health club offers the following programs: *Yoga, Cardio, Zumba and Weight lifting*. All the trainers in the club can train for the only specified programs. The operating hours of the club are from 6 am to 9 pm. Each program requires a 1-hour slot and a member can undertake only one program in a day. A trainer can train only one member during the one-hour slot. The owner collects the requests from all the members as per their choice of program, the preferred time slot, preferred trainer and schedules a 1-hour slot for each of the members depending on the availability of the trainers. The trainers are paid the remuneration for every 1-hour training session conducted by them. Implement this system using Linked Lists only as the data structure to store and maintain the records.

The record for members must have the following fields: Member-ID (should be a 4-digit randomly generated number), name, age, gender, programs enrolled for (a member can enrol for more than one program), fees paid for the corresponding program.

The record for trainers must have the following fields: Trainer-ID, name, list of programs which can be trained by the trainer, remuneration paid for 1 training session.

Your implementation must have the following functions:

`1. Add_member()`
Input: all the fields related to a member listed above (Member-ID should be randomly generated 4-digit number).
Output: Display a message after successfully adding a member.
Note: Records of members have to be maintained in a sorted manner based on key as their member-ID.

`2. Add_trainer()`
Input: all the fields related to a trainer listed above.
Output: Display a message after successfully adding a trainer.
Note: Records of trainers have to be maintained in a sorted manner based on key as their Trainer-ID.

`3. Schedule_slot()`
Input: list of members with their requested program, preferred time slot, preferred trainer (only 1) & the timestamp at which the request is made, and the list of trainers available in the club (note: All the trainers from the trainers' list may not be available).
Output: list of members assigned to trainers along with their time slots during operating hours and the list of non-assigned members, if any.
Note: The slots are assigned on a First-come-First-serve basis as per the request timestamp. If none of the trainers is free during the preferred time, then the member is assigned to the earliest available slot after the preferred time slot. As per the covid-norms set by the

government and the floor capacity of the club, not more than 10 members can be present inside the club at any time. The trainers cannot be assigned for a program they don't train for. If a member requests for more than one program in a day, then the later request should be disallowed.

`4. Print_hour_wise_list():`
Input: No input
Output: Hour/Slot-wise list of members & trainers assigned grouped by every program.

`5. Delete_member():`
Input: Member-ID
Output: Display a message after the successful deletion of the record.

`6. Delete_trainer():`
Input: Trainer-ID
Output: Display a message after the successful deletion of the record.

`7. Search_member():`
Input: Key field to search and key-value
Output: Display complete record based on key-value.

`8. Search_trainer():`
Input: Key field to search and key-value
Output: Display complete record based on key-value.

`9. Print_member_list()`
Output: Display the records of all members.

`10. Print_trainer_list()`
Output: Display the records of all trainers.

`11. Print_sorted_remuneration_list():`
Output: Display the list of trainers in a sorted order according to their remuneration earned (highest to lowest) in a day.

`12. Print_intersection_list():`
Input: list of members same as in function 3.
Output: list of members who have been assigned their preferred trainer.

The program should have menu-driven options for the owner and it should be able to accept all the necessary input. You are allowed to make use of additional functions as per your need. The program must be able to dynamically handle the database.

Problem for R2 Batch (Enrolment Numbers BT20CSE031 to BT20CSE060):

Book Access Management**:** Define a list of all different books (***book_list***) stored in a library. The list contains the following information corresponding to each book:

- Title, author/s, subject, number_of_copies_issued, number_of_copies_available

Define a student list (***request_queue***) containing information about those who are currently asking for books. This list is having information like:

- Name_of_the_student, title_of_the_book_asking_for

If a student is asking for more than one book at a time, create separate node corresponding to each request. The books will be given on first-come-first-serve basis till the availability of the book. If one student raises request for more than one book and already one book is given to him; he should be moved to the last position (lowest priority) of the ***request_queue***. A student can borrow maximum 3 books at a time. A student, who has not returned book/s on time, is not allowed to issue another book though he may raise a request.

Create another list of students (***borrower_list***) who already borrowed books. The nodes of this list contains the following information:

- Name_of_the_student, title_of the_book, date_of_issue/return

Students can keep a book for maximum 15 days. If a student does not return a book on or before return date, he is called a defaulter.

Create 3 separate linked lists named as ***book_list, request_queue*** and ***borrower_list***. Assume that no two different books belong to the same subject and having different author/s have the same title.

1. Display the name of the students who will get books.
2. Write a function to find the book in most demand? The demand of a book can be defined as the sum of the number of books issued and the number of books requested.
3. Within first 3 days how many books can be given to the students? Write a function for that.
4. Sort the ***borrower_list*** according to the number of books issued in descending order.
5. Sort and display the title and author's name for all books of a particular subject on the basis of number_of_copies_available.
6. Remove the defaulters from the ***request_queue*** and insert them into a new list named as ***defaulter_list***. Display the ***defaulter_list***.
7. Display the names of the students who have already borrowed books and are asking for other books.
8. Display the names of the requested books whose copies are available.
9. Display the title of all the books which have not been issued by anyone.
10. Display the name of the student/s who has requested for the maximum number of books.

Problem for R3 Batch (Enrolment Numbers BT20CSE061 to BT20CSE090):

A phonebook can be implemented as a linked list of data structures where each node may represent first name, last name and mobile number and you may also add addition nodes like e-mail, office no., address, etc for every contacts. You can make personal contact (name, mobile number, email, type) and professional contact list (name, mobile number, email, office number, address,institute/office, type). Keep the list sorted based on name. If names in two records are the same, then such records are to be ordered based on mobile number (by considering its integer value).

You need to provide a field according to which personal and professional contacts would be separate. (Type: personal/professional). In addition, such a phonebook should keep track of number of phone calls made to different contacts, the date and the duration of the calls.

The following functions/operations need to be implemented:

1. Insert/create contact:
    a. A new contact to be inserted in the list. Make sure that there is no restriction for the contact with same name.
    b. i/p: all the field related to insertion on a new contact
    c. o/p: A message showing that contact is created
2. Edit:
    a. Edit option should be provided to modify the details.
    b. i/p: all the fields that can be modified
    c. o/p: message showing that contact in modified
3. Delete:
    a. contact can be deleted as a whole
    b. i/p: field to search for contact that is to be deleted
    c. o/p: message showing contact is deleted
4. Search:
    a. provide searching contact via name, mobile number and any other if you like
    b. i/p: provide options to search using name and mobile no.
    c. o/p: contact searched
5. Sort:
    a. Sorting in ascending /descending, professional/personal
    b. i/p: Given unsorted list of records, provide the options for sorting i.e. using switch cases like ascending, descending.
    c. o/p: sorted list according to the choice of user

6. Display all:
    a. Display personal list: only personal contacts to be displayed
    b. Display professional list: only professional contacts to be displayed

    c. i/p: option asking to show contacts
    d. o/p: list as per option selected

7. Union:
    a. merge two linked lists sorted based on name and mobile number
    b. i/p: Two linked lists sorted based on name and mobile number
    c. *o/p: A single linked list with union of records in two given linked lists, sorted based on name and mobile number*

8. Remove duplicates:
    a. i/p: enter the list of records sorted based on name/mobile-number
    b. o/p: linked list without duplicates

9. Design a linked list that stores phonebooks of different people, that is we have to design linked list of linked lists. For each phonebook owner, we can keep minimum details like name, mobile number, address. From such a database, we need to display the names of people in the decreasing order of total time duration they have spoken to their own contacts.

Problem for R4 Batch (Enrolment Numbers BT20CSE091 to BT20CSE128, + ex-students):

Implement the following system using a linked list of structures where every node of that list represents a record of passenger name, passenger id, boarding train, boarding station, travelling class(Sleeper, 3AC, 2AC, 1AC), destination station, train id, Seat number (bogie number/seat number), and any other field you think that would be useful to passengers. You can also take a confirmation from the passenger whether upgrade of travel class is desired.

The passenger id can be thought as a key in the list and will represent a unique record in the list. The records should be always kept sorted according to the train id so that passengers boarding the same train have their data together.

You can assume number of bogies of each class and number of seats in each bogie.

Write the following functions :

-insert

- Insert a list of passengers and their details for the reservation
- I/p parameters: Reservation request that includes a list of passenger names, passenger ids, boarding train, boarding station, travelling class(Sleeper, 3AC, 2AC, 1AC), destination station, train id
- O/P: Reservation done successfully, partially or the reservation failed.
- Note – The set of passengers in a single reservation request should be allocated seats together. If all of them cannot get the seats together, then they need to be accommodated as close to each other in trains, that is, their bogie/seat numbers should be as close to each other.

-delete

- Deletes an element if the passenger cancels the reservation.
- I/p parameters: deleting all records of that particular passenger id.
- O/p: If node gets deleted print Reservation cancelled successfully or if it gets failed then print Reservation Cancellation failed.

-getListDestination

- Get the list of passengers having the same destination station and same train id

-SortByTravelDate

- Input – Passenger id
- Output – Display the list of destination stations for a particular passenger as per the dates of the travel.

-SortTrains

- I/p parameters: The train database that is the linked list with passenger data as given
- O/p: Display the train number and the travel date in the sorted order of number of passengers on the train.

- PromotePassengers

- Input – Train id and date of travel
- Output – For all the passengers on the train with train id and a particular date, passengers can be promoted to next travel class (Sleeper -> 3AC -> 2AC ->1AC) if seats are available. Passengers who had given consent for promotion to next class are considered in the order of their date of booking. Note that if 2AC passenger is promoted to 1AC, his/her 2AC seat becomes vacant and can be occupied by another passenger from lower class under promotion.