CSL 214 : Data Structures and Program Design II
H/W Assignment-2

1. Announcement Date: **30-03-2023**

2. Due date: Submit online by **10:00 AM on Monday 10th April, 2023**

3. The assignment can be done either individually or in a group of 2 students. In case of group of 2, both the partners in the group should belong to the same practical batch.

4. For this assignment, Tree data structure is required to be used. Hence, choose Trees (either AVL tree OR B-tree OR B+tree) for any storage requirement you may have. Use of file operations to store/retrieve persistent data, would be given more credits.

5. No Copying / sharing of code is acceptable. If any such case is identified, the original author(s) and person(s) who has copied, both will be penalised equally and zero marks will be awarded.

6. 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-2: Your enrolment numbers.

7. The evaluation via will take place in the week of 10th April 2023 to 14th April 2023. The evaluation is not considered complete unless the viva of the project groups during lab hours takes place. All students are required to attend their lab hours without fail in the week of 10th April 2023.

Problem for R1 Batch (Enrolment Numbers BT21CSE001 to BT20CSE025):

To conduct any examination, an eligible candidate database should be maintained by the organization. For example, consider the semester examination of an education institute. Let for the current semester, the examination will be conducted for 2^{nd}, 4^{th}, 6^{th} and 8^{th} semester students for all departments. There are five departments namely, Computer Sc. & Engg., Electronics & Comm., Electrical, Mechanical and Civil. The corresponding department codes are CS, EC, EE, ME and CV respectively. Also consider that every student has registered for 5 different courses belonging to his/her department.

Create a database of ***general student records*** comprises of the following information:

- Name of student
- Department code
- Year/Sem
- Roll number (unique key across dept. and year)
- Number of classes conducted for each of the subjects registered.
- Number of classes attended by the student for each of the subjects registered.

To track the attendance eligibility, only one subject (for which maximum number of classes were conducted) is considered for every semester and every department, and is denoted as MCC. The percentage of attendance is computed as below-

The attendance of the student, who has maximum number of classes attended for MCC, is considered as 100%. With respect to this, % of attendance for other students as required is to be computed using his/her attendance for MCC.

Create four attendance record databases, each database corresponds to one semester, to maintain the attendance of all student in the particular semester irrespective of their departments. The following information are to be kept in the attendance database:

- Roll number
- Department
- Attendance : Number of class attended for MCC and % attendance

Create fee status records database to maintain the record of the fee status for all students. The database contains the below information:

- Roll number
- Fee status (pending/clear)

Create an applicant records database to track the record of the students who have applied for the examination. The database stores the following fields:

- Name of the student
- Roll number
- A/NA (A stands for applicant, NA stands for non-applicant)

Create the ***general student records*** database by reading the data (either from the users or through file reading operations).

Write efficient C code to accomplish the following operations.

1. Sort the ***general student record*** database based first on semester, and then followed by department-name, and then the roll number.
2. Display ***year wise*** and then the ***department-wise*** names of the students who did not apply for the examination.
3. A student is eligible to sit for the semester examination if his/her attendance is more than 75% for MCC, fees status is clear and he/she applied for the examination. Create a list of all eligible students for the semester examination. The list of students is to be based first on semester, and then followed by department-name, and then the roll number.
4. Create a list of students having attendance <=75%. based first on semester, and then followed by department-name, and then the roll number.
5. Delete a student record. Input is roll number.
6. Print the name of the students whose attendance is >75% for the respective MCC but their fee status is pending.
7. Create a list of defaulter students. A student is called as defaulter either his/her attendance for MCC is <=75% or his/her fee status is pending. Find the name of the department in which maximum number of defaulters belong to.
8. Range-search students based on the roll number. Inputs are two roll numbers enrol1 and enrol2, and the output is the list of students having roll numbers *larger* than *enrol1* and *less* than *enrol2*.

Note: The information regarding the fields corresponding to each database (student records, attendance records, fee status records and applicants' records) should be maintained strictly. It is desirable to take the input through file handling. The program should be executed by taking the records of at least 100 students as input (minimum five students for every year and department).

Problem for R2 Batch (Enrolment Numbers BT21CSE026 to BT21CSE050):

HungryApp : HungryApp should let you search for and discover restaurants to eat out at or order from. The app lets user to browse through restaurant menus to decide where one wants to eat. A restaurant is identified by name as well as address which can be treated as a unique key. A user is identified by either User Id or Phone No**.** The food-delivery agent is identified by his agent-id or his phone number.

Right now, we will like to develop the App not as a mobile app but just as a desktop application. To develop the application, maintain the following databases.

1. Restaurant database – Records of restaurants with details like address/area, no-of-seats, special facilities, menu etc,

2. Agents-database - Each agent identified by agent-id, name, phone No., currently-accumulated-commission etc.

3. User-database, each user identified by user-id, name, address, phone no, etc. For every user, a record of all previous orders is also kept.

Following functions are required to be provided.

1. **Different Search() functions to search for eating locations;**
   - Based on category (restaurant, cafe, pub etc)
   - Based on category of food or cuisine (north Indian, south Indian, continental etc)
   - Based on area (Name of area and its nearby areas). Note that you can keep information which tells which other areas of city are near the given area.

2. **Order (customer name, address, Phone No., Eating-joint, Menu)**;
   Each order should be added to the global database of orders which maintain all the orders yet to be delivered. This function allocates an available agent to this particular order. The output of this function should be order details along with agent details. You can decide the sorted order you want to store the global database of orders so that it is easy for different operations.

3. **Delivery()** : This function is to be executed on the actual delivery and the order from global order database is removed which also frees the respective agent. It also adds 10% commission to the particular agent's currently-accumulated commission.

4. **CancelOrder()** : It cancels the order, removes it from global database of orders and frees the agent.
5. **FindFavoriteFoodsOfUser(user-id) :** For the given user-id, find top 3 favorite food items based on previous orders of the last 3 months.
6. **FindFavoriteRestaurants(int N) :** Finds the top-3 restaurants that have got maximum orders in the last N days.
7. **FindFavoriteFoodsAcrossRestaurants(int N) :** Finds the top-3 ordered most favorite food items in the last N days.

8. **Range-Search –** Create/print a list of current orders for users having user-id between *user-id1* and *user-id2*.

In addition, following print operations to be supported.

- Print agents database and their details
- Print live orders from global database
- Print area wise agents.
- Print restaurant details for the given restaurant name and address.

Problem for R3 Batch (Enrolment Numbers BT21CSE051 to BT21CSE075):

We need to design a data structure for music player. A tree database can store the database of all music listings. Each music listing will have attributes like singer, lyricist, album/film, composer, genre, duration of the song and any other you may choose. We need to define following operations:

1. Write a function, InsertSong(), which reads a song from user and insert into a tree database of structures.
2. Write a function removeDuplicates() to remove the duplicate entries from the records.
3. Write a function to printPlaylist() which prints the records in the tree database in ascending as well as in descending order.
4. Write a function to DeleteSong() to delete the particular song details from the tree-database as per the song_name and also as per the name of the artist provided by the user and deletes all the entries with that artist.
5. Create a play-list (using a tree data structure) based on the attribute and its value given. For example, if "singer" attribute is provided and "Kishor Kumar" is the name of the singer, then a new play-list is created containing all the songs belonging to Kishore Kumar. The songs for the same singer to be sorted based on lexicographic order of the songs. Multiple attributes and their values can also be provided to create the play-list. For example if attributes given are "singer" and "film" and corresponding values provided are "Kishor Kumar" and "Sholay", then all songs belonging to Kishore Kumar in Sholay are added into the play-list and likewise.
6. (Dis)Play a song in a play-list. In our setting this will mean, given a play-list as input, show/print all songs in the play-list indexed with serial numbers. User can select a serial number of the song, and data corresponding to the selected song is required to be displayed. At the same time, name of the previous as well as next song is also displayed. User is provided an option to select "Next" or "Prev" song. By selecting one of the options either "Next" or "Prev", respectively data for next and previous song is displayed.
7. Given a play-list and the serial number of the song, display details of the song. At this point, given an integer offset "k" and direction "up" or "down", display the details of k^{th} previous or k^{th} next song, from the current song, respectively, in the play-list.
8. Given a play-list and the serial number of the song, display details of the song. Provide an option to the user to select one or many attributes of this "current" song and create a play-list for these attribute values from the original database.
9. Filter the playlist. Given a play-list and an attribute value, filter all songs in the playlist matching the attribute value. For example, if play-list is for "Kishore Kumar" and attribute value is given for genre as "comedy", all songs in the current playlist with genre as "comedy" will be chosen.
10. Shuffle to something else. Take a set of attributes and their values as inputs. Randomly select any song from the original database matching all the attribute values provided. Mark it using FLAG so that it doesn't get a chance again till all other songs in the playlist are played randomly.

11. Range-search – Search songs belonging to singers whose names are in-between *singer-name-1* and *singer-name-2*, in lexicographic sorting sense.

Problem for R4 Batch (Enrolment Numbers BT21CSE076 to BT21CSE100):

You need to develop a system for registration, allotment of classrooms for lectures as well as examination allotment for students. Maintain one database for students' data. Every student has registered for five courses. Maintain information such as name of student, enrolment number, courses taken etc. Every course will also have its record of students taking that course, slot assigned to that course, maintained separately. Multiple courses (across semesters and also for the same semester) can be assigned to the same slot.

Maintain another database of classrooms. For every room, maintain the number of seats in that room. Also assume that there are 4 rows in each room and each bench can provide seats for 3 students. The number of benches per row would depend on the class capacity.

Implement features/functions such as :

1. Adding a student to the student-database. When adding a student to the student-database, verify that a student cannot take 2 courses in the same slot.
2. Deletion of a student from the student-database. When deleting a student, ensure that the student record is also deleted from the courses taken by that student.
3. Adding a course to the course database. When deleting a course, ensure that no student has taken that course.
4. Allotment of classrooms for lectures. Allot every course to a classroom such that the capacity of the room is larger than the number of students registered for the course. Allot the classrooms in the most optimal way such that minimum number of classrooms are utilised. Two courses in the same slot to get different rooms. Also as much as possible the same room is to be used for consecutive slots for students of the same class/semester. If it is not possible, then different rooms can be utilised. Detect the situation if a room cannot be allocated to some course in a particular slot. In such case make sure that least number of courses are not allocated the rooms and these courses together have as less number of students as possible.
5. Allotment of classrooms for exams. Seats in the classroom can be numbered based on row-number, bench-number etc. When assigning seats in the examination hall, ensure that no students of the same course can be assigned adjacent seats.
6. Print the list of students and the exam seat allotted to each student for a given slot.
7. Range-search – Search and print list of courses which have course-ids between *course-id-1* and *course-id-2*.

Problem for R5 Batch (Enrolment Numbers BT21CSE101 to BT21CSE129 + ex-students):

Assignment Management Software System

You need to develop a system for managing student assignments and submissions. Assume that students will form groups each containing 2 students. The teacher may decide to give the same or different assignments to different groups. The teacher may evaluate the assignments offline, also teacher can conduct viva for each group, but may decide to give different marks to students in the same group.

One database to be maintained for students. Each student record contains student-id, name, group-partner, group-id, assignment-name/id, deadline, status-of-assignment (submitted/not-submitted/evaluated), offline-evaluation-marks, viva-marks.

Another database is maintained for assignment records. Each record contains assignment-id, topic-name, status (declared/due-date-over/evaluated). The same assignment may be given to multiple student groups. Each assignment-record in turn can have a separate database inside, each record containing student-group-id, status (declared/submitted/not-submitted/evaluated), marks given.

Following operations are to be defined :

1. Insert a student record in the student-database. After every insertion of a student-record, appropriate changes/insertion needs to be done in the assignment-database also.
2. Insert assignment-record in the assignment-record-database. The insertion function can take a list of students and allocate the assignment to multiple student groups, thus resulting into insertions in student-database as well.
3. Change status of assignment and student database based on different events
    a. Student-group submitting the assignment
    b. Assignment for one specific student group is evaluated. Once a particular assignment is evaluated for all student-groups to whom the assignment was given, the status of the assignment itself changes to “evaluated”.
4. Print the details of assignments that are declared but not evaluated.
5. Print the details of student groups who have not submitted the assignments even if the due-date is over.
6. Print the details of student groups for whom the assignment is yet-to-be-evaluated even though they have been submitted. In case viva is remaining to be taken, that is explicitly to be mentioned.
7. For a given assignment id, print the details of student-groups, in the decreasing order of marks they have received in the assignment.
8. Print student groups receiving top marks in each of the assignment given.
9. Range-search – Search and print the details of assignments having assignment-id between *assignment-id-1* and *assignment-id-2*.

You can decide to create a menu-driven program (not compulsory, but extra credits may be given) where students and the teacher can log in (password not mandatory) and perform all the above operations.