Mini Project Problem Statements
Last Date for submission: 19 Nov. 2021

- For your reference some problem statements are given below. But you are not restricted to these problems statements.
- You are free to choose any problem statement of your choice but make sure your solution covers most of the C programming concepts **(input/output statement, Conditional statements, looping constructs, Methods/Functions/recursion, Linked list, Structure/ Union, Arrays and if possible file handling)** which we have studied in the class.
- You are allowed to refer the online resources and use some function from there but make sure you have understood the code and then used the same in your project.
- No coping/sharing of code across the groups. If found copied, NO marks will be awarded to assignments for all such groups including original authors.
- Evaluation will be done during (Will inform you later). Please make sure that you will be ready with your solution.
- You need to send your source files through mail to ta1@coed.svnit.ac.in, ta2@oed.svnit.ac.in and ta12@oed.svnit.ac.in.
- Please mark your subject line as DS-Project Assignment: Your Registration Number.

Note: As some of the concept which you required for completion of this assignment are not yet covered in the class. So, that concept will be covered once our regular classes will be started. Till that time try to complete the part of assignment on the basis of your understanding of whatever we have covered and try to understand some of the concept whichever possible from internet resources.

Employee Management System:

A Employee Records Management System can be implemented as an array of structures, where each element of the array represents an employee record and each record has fields like employee name, salary, date of joining, Date of birth, level (A, B, C), designation, and any other field you may want.

The combined pair of joining date and date of birth of the employee can be thought of as a key in the list and represents a unique record in a single list. Hence, we required joining date and date of birth to access the details of the particular employee.

The list of records should always be kept sorted (after any operation) on the basis of key.

Write following functions and a program that lets user to use any of these operations.

-insert

- Inserts (if the entry is not present in the list) or updates an element (if the entry is present in the list)
- *I/p parameters: employee name, employee id, salary, date of joining, Date of birth, level (A, B, C), designation.*
- *O/p – Entry should be added/updated appropriately. The return parameter can specify if the insert operation is successful or not.*

-delete

- *Deletes an elements*
- *i/p parameters:*
    - *list (array of records from which the entry needs to be deleted)*

    - *employee id and employee name of the employee whose records need to be deleted.*
- *o/p parameters: To know the operation is successful or not you have to print the appropriate message.*

-getNumEmployees
- *i/p parameters: list (array of records)*
- *o/p: number of employees of each level separately.*

-isEmpty
- *i/p parameters: list (array of records)*
- *o/p: To know if the list is empty or not*

-isFull
- *i/p parameters: list (array of records)*
- *o/p: To know if the list is fully occupied or not*

-Updateemployeestatus
Parameters: list2 (employee id, employee name, level, salary)
i/p paraments: list 2
o/p: Updates the original list. The Output must be sorted on key.
-getLongestEmploymentPeriod
i/p parameters: list1
o/p : employee with max (Joining Date – Retirement Date)
Note: Retirement age is 60.
-list Unique
- *i/p parameters – A list of record which contains duplicate entries (with same employee name and employee id)*
- *Output – Remove duplicate entries if present. Retain the occurrence with highest level, in case of duplicates.*

-listUnion
- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains union/merge of two lists.
- The union (list3) is again should be sorted on key.
- In case, an entry is present in both the lists, entry in the first list is taken and the one in the second list is dropped.

-listIntersection
- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains intersection of two lists.
- The list3 should be sorted on key.
- For an element having its key in both the lists, retain the entry with highest level.

- listDifference
- Parameters - list1, list2,list3

- i/p parameters – list1 and list2
- o/p parameters – list3 = list1-list2; (similar to set difference operation)
- Output contains those elements whose keys are present in list1 but not in list2. The output must be sorted on the key.

-listSymmetricDifference

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
    - Whose keys are present in the list1 but not in list2 AND whose keys are present in list2 but not in list1
- Output must be sorted on key.

Flight Management System:

A Flight Records Management System can be implemented as an array of structures, where each element of the array represents a flight record and each record has fields like flight name, flight id, flight capacity, flight arrival time, flight departure time, flight class (VIP, VVIP, public), and any other field you may want.

The combined pair of flight id and arrival time of the flight can be thought of as a key in the list and represents a unique record in a single list. The list of records should always be kept sorted (after any operation) on the basis of key.

Write following functions and a program that lets user to use any of these operations.

-insert

- Inserts (if the entry is not present in the list) or updates an element (if the entry is present in the list)
- *I/p parameters: flight name, flight id, flight capacity, flight arrival time, flight departure time, flight class (VIP, VVIP, public).*
- *O/p – Entry should be added/updated appropriately. The return parameter can specify if the insert operation is successful or not.*

-delete

- *Deletes an elements*
- *i/p parameters:*
    - *list (array of records from which the entry needs to be deleted)*
    - *flight id of the flight whose records need to be deleted.*
- *o/p parameters: To know the operation is successful or not you have to print the appropriate message.*

-getNumFlights

- *i/p parameters: list (array of records)*
- *o/p: number of flights in the list.*

-isEmpty

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is empty or not*

-isFull

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is fully occupied or not*

-UpdateFlightStatus
Parameters: list2 (flight id, flight name, delay time, status (Delay, On time, Cancelled))
i/p paraments: list 2
o/p: Updates the original list. The Output must be sorted on key.
-getlongeststay
i/p parameters: list1
o/p : flight with max (arrival time – departure time)
-list Unique

- *i/p parameters – A list of record which contains duplicate entries (with same flight name and flight id)*
- *Output – Remove duplicate entries if present. Retain the occurrence with minimum delay, in case of duplicates.*

-listUnion

- *Parameters - list1, list2,list3*
- *i/p parameters – list1 and list2*
- *o/p parameters – list3*
- Takes list1 and list2 as inputs and creates list3 as output that contains union/merge of two lists.
- The union (list3) is again should be sorted on key (flight id and flight name)
- In case, an entry is present in both the lists, entry in the first list is taken and the one in the second list is dropped.

-listIntersection

- *Parameters - list1, list2,list3*
- *i/p parameters – list1 and list2*
- *o/p parameters – list3*
- Takes list1 and list2 as inputs and creates list3 as output that contains intersection of two lists.
- The list3 should be sorted on key (flight id and flight name)
- For an element having its key in both the lists, retain the entry with minimum delay time.

- listDifference

- *Parameters - list1, list2,list3*
- *i/p parameters – list1 and list2*
- *o/p parameters – list3 = list1-list2; (similar to set difference operation)*
- Output contains those elements whose keys are present in list1 but not in list2. The output must be sorted on the key.

-listSymmetricDifference

- *Parameters - list1, list2,list3*
- *i/p parameters – list1 and list2*
- *o/p parameters – list3*
    - Whose keys are present in the list1 but not in list2 AND whose keys are present in list2 but not in list1

Output must be sorted on key.

Inventory Management System:

A Inventory Management System can be implemented as an array of structures, where each element of the array represents a computer record and each record has fields like computer id, date of purchase, date of manufacture, warranty period, processor (I3, I5, I7), life period (returned off date), and any other field you may want.

The combined pair of date of purchase and computer id of the computer can be thought of as a key in the list and represents a unique record in a single list.

The list of records should always be kept sorted (after any operation) on the basis of key.

Write following functions and a program that lets user to use any of these operations.

-insert

- Inserts (if the entry is not present in the list) or updates an element (if the entry is present in the list)
- *I/p parameters: computer id, date of purchase, date of manufacture, warranty period, processor class (I3, I5, I7), life period (returned off date).*
- *O/p – Entry should be added/updated appropriately. The return parameter can specify if the insert operation is successful or not.*

-delete

- *Deletes an elements*
- *i/p parameters:*
    - *list (array of records from which the entry needs to be deleted)*
    - *computer id of the computer whose records need to be deleted.*
- *o/p parameters: To know the operation is successful or not you have to print the appropriate message.*

-getNumComputers

- *i/p parameters: list (array of records)*
- *o/p: number of computers of each level separately.*

-isEmpty

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is empty or not*

-isFull

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is fully occupied or not*

-UpdateProcessor

Parameters: list2 (computer id, processor class, price)

i/p paraments: list 2

o/p: Updates the original list. The Output must be sorted on key.

-getLongestReturenedOffPeriod
i/p parameters: list1
o/p : computer with max (Returned off Date –Purchase Date)
-list Unique

- *i/p parameters – A list of record which contains duplicate entries (with same computer id)*
- *Output – Remove duplicate entries if present. Retain the occurrence with highest processor class, in case of duplicates.*

-listUnion

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains union/merge of two lists.
- The union (list3) is again should be sorted on key.
- In case, an entry is present in both the lists, entry in the first list is taken and the one in the second list is dropped.

-listIntersection

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains intersection of two lists.
- The list3 should be sorted on key.
- For an element having its key in both the lists, retain the entry with highest processor class.

- listDifference

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3 = list1-list2; (similar to set difference operation)
- Output contains those elements whose keys are present in list1 but not in list2. The output must be sorted on the key.

-listSymmetricDifference

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
    - Whose keys are present in the list1 but not in list2 AND whose keys are present in list2 but not in list1
- Output must be sorted on key.

Process Management System:
A Process Management System can be implemented as an array of structures, where each element of the array represents a process record and each record has fields like process id, run time, priority, class of priority (A, B, C), and any other field you may want.
Note: Each process has a class of priority field; each class can have priorities as positive integer value. Lowest integer considered as the higher priority exclude 0.
The combined pair of priority and priority class of the process can be thought of as a key in the list and represents a unique record in a single list. The list of records should always be kept sorted (after any operation) on the basis of key.
Write following functions and a program that lets user to use any of these operations.
-insert

- Inserts (if the entry is not present in the list) or updates an element (if the entry is present in the list)
- *I/p parameters: like process id, run time, priority, class of priority (A, B, C).*
- *O/p – Entry should be added/updated appropriately. The return parameter can specify if the insert operation is successful or not.*

-delete

- *Deletes an elements*
- *i/p parameters:*
    - *list (array of records from which the entry needs to be deleted)*
    - *process id of the process whose records need to be deleted.*
- *o/p parameters: To know the operation is successful or not you have to print the appropriate message.*

-getNumProcess

- *i/p parameters: list (array of records)*
- *o/p: number of process in the list.*

-isEmpty

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is empty or not*

-isFull

- *i/p parameters: list (array of records)*
- *o/p: To know if the list is fully occupied or not*

-UpdateProcessPriority
Parameters: list2 (process id, Priority Class, Priority)
i/p paraments: list 2

o/p: Updates the original list. The Output must be sorted on key.
-runProcess
i/p parameters: list2 (n)- n represents top n priority process
o/p : Update original list
Note: The process which is executed will not be present in the updated list. Consider the class also while updating.
-list Unique

- *i/p parameters – A list of record which contains duplicate entries.*
- *Output – Remove duplicate entries if present. Retain the occurrence with maximum class priority, in case of duplicates.*

-listUnion

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains union/merge of two lists.
- The union (list3) is again should be sorted on key.
- In case, an entry is present in both the lists, entry in the first list is taken and the one in second list is dropped.

-listIntersection

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
- Takes list1 and list2 as inputs and creates list3 as output that contains intersection of two lists.
- The list3 should be sorted on key.
- For an element having its key in both the lists, retain the entry with maximum class priority.

- listDifference

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3 = list1-list2; (similar to set difference operation)
- Output contains those elements whose keys are present in list1 but not in list2. The output must be sorted on the key.

-listSymmetricDifference

- Parameters - list1, list2,list3
- i/p parameters – list1 and list2
- o/p parameters – list3
    - Whose keys are present in the list1 but not in list2 AND whose keys are present in list2 but not in list1
- Output must be sorted on key.

Some other project Title you can chose any project of your choice:

- **Library Management System**
- **Hotel Management System**

- **Student Information System**
- **Hospital Management System**
- **Vehicle Management System**
- **Payroll Management System**
- **Car/bike Showroom Management System**
- **Railway Reservation Management System.**
- **Shopping Mall Management System**
- **Movie Ticket Booking Management System**
- **Bank Management System**