Ex : 5

Title: Design, Develop and Implement a menu driven Program in C for handling change of branch.

Problem Description: After 1st year, students can apply for change of branch and are generally selected based on their CGPA. Assume that among the selected students, some may change their mind. Some existing students also may choose to leave and join another department. Display the complete student data with regular and change of branch students, Limit your program to only your department.

Method: Make use of arrays and linked lists.

Theory Reference: Module 2

Explanation:

1. **Data Structure:**
   - The list of students can be implemented as a singly linked list.
   - To represent a node of a singly linked list in C, a structure can be used that has four data members usn, name, mode and next where:

     usn: indicates the USN of the student.

     name: indicates the name of the student.

     mode: indicates the mode of the student(Regular/Lateral/COB/COC)

     next: is a pointer that will store the address of the next node in the sequence.

2. **Singly Linked List**

   A singly linked list is a type of linked list where only the address of the next node is stored in the current node along with the data field and the last node in the list contains NULL pointer. This makes it impossible to find the address of the particular node in the list without accessing the node previous to it. So we can only access the data sequentially in the node.

   Basic list operations:

   i. Create a list – Creates the first node in the list called start.

ii. Insert at the end of the list – inserts a node to the end of the list.

iii. Delete a node from the list – deletes the node whose usn == key, the value of key is entered by the user.

iv. Display the list – Displays the USN and mode (Regular/Lateral/COC/COB) of all the students in the class

3. **Freeing Memory:** The dynamically allocated memory is freed using free() to avoid memory leaks.

Algorithm:

Step 1: Initialize

- Define a self-referential node.
- Create the first node in the list called start (by dynamic memory allocation function, malloc()) and assign the USN of the first student.

Step 2: Insert at the end of the singly linked list.

- Create a new Node.
- If the list is empty, update the start pointer to be this new node and then return.
- Otherwise traverse till the last node of the singly linked list.
- Connect next pointer of the last node to the new node.

Step 3: Delete a node in the singly linked list, whose usn == key.

- Ensure that the Head of the linked list is not NULL; if it is, the list is empty, so return.
- If the singly linked list has only one node, delete the head node and point the head pointer to NULL.
- Traverse till the node whose usn == key is found.
- Use a second pointer to track the previous node to the current node.
- Connect the next node of the current node as the next node of the previous node.

- Delete (free) the current node represented by the temporary pointer.

Step 4: Display Operation

- Check if the HEAD of the singly linked is is NULL or not. If NULL return back.
- Set a temp pointer to the head of the singly linked list.
- While temp pointer != NULL:
- Print temp->data.
- Move temp to temp->next

Step 5: Main Function

1. **Initialize the singly linked list.**
   - Ask the user to input the number of students.
   - Create the list with the details of the first student.
   - Add the remaining students by inserting at the end of the list.
2. **Loop through the menu:**
   - Print the menu and ask the user to enter a choice.
   - Based on the user's choice, call the corresponding function:
     - Choice 1: Call insertend() to initialize the list of regular students.
     - Choice 2: Call display() to print the details of all the students in the class.
     - Choice 3: Call deletekey() to delete the details of the student whose USN matches the key, entered by the user.
     - Choice 4: Call insertend() to add the details of Lateral/COC/COB students.
     - Choice 5: Exit the program and free the allocated memory.
   - If the user enters an invalid choice, print "invalid choice."

Step 6: Terminate

- When the user chooses to exit, free the allocated memory for the stack and terminate the program.