COM2002 INTERMEDIATE PROGRAMMING
2024 – 2025 SPRING

Laboratory Week: 24 – 28 April 2025

Topic : Advanced Uses of Pointers

Program-1 : The Printing a One-Month Reminder List (remind2.c)

Definition : The program is a modification of **the Printing a One-Month Reminder List (remind.c) program** which prints a one-month list of daily reminders. The original remind.c program stores reminder strings in a two-dimensional array of characters. In the new program, the array will be one-dimensional; its elements will be pointers to dynamically allocated strings.

Advantages of switching to dynamically allocated strings:

- Uses space more efficiently by allocating the exact number of characters needed to store a reminder.
- Avoids calling strcpy to move existing reminder strings in order to make room for a new reminder.

Program-2 : Inventory (inventory2.c)

Definition : The program is a modification of the parts database program of Laboratory 6 (the original program), with the database stored in a linked list this time.

Advantages of using a linked list:

- No need to put a limit on the size of the database.
- Database can easily be kept sorted by part number. In the original program, the database wasn't sorted.

The part structure will contain an additional member (a pointer to the next node) and *inventory* will point to the first node in the list.

Most of the functions in the new program will closely resemble their counterparts in the original program. *find_part* and *insert* will be more complex, however, since we'll keep the nodes in the *inventory* list sorted by part number.

In the original program, *find_part* returns an index into the *inventory* array. In the new program, *find_part* will return a pointer to the node that contains the desired part number. If it doesn't find the part number, *find_part* will return a null pointer. Since the list of parts is sorted, *find_part* can stop when it finds a node containing a part number that's greater than or equal to the desired part number. When the loop terminates, we'll need to test whether the part was found.

The original version of *insert* stores a new part in the next available array element. The new version must determine where the new part belongs in the list and insert it there. It will also check whether the part number is already present in the list.

Try to write *erase* function: The codes e (erase) will be used to represent delete operation. Prompts the user to enter a part number. Prints an error message if the part doesn't exist; otherwise, removes the part from the database.