Assignment 2 – Dynamic Task Management

In many real applications, data items are inserted and removed frequently, and the total number of items can grow or shrink dynamically. An **array** may waste space or require expensive shifting/resizing. A **linked list** solves this by storing items as nodes connected using pointers, allowing efficient insert/delete operations. In addition, such operations are often tracked for auditing, recovery, and analysis purposes. Examples include transaction logs, undo features, and activity histories.

You are required to develop a **menu-driven program** using a **singly linked list** to manage records.

Each record represents a Task in a personal task tracker:

- int taskID (unique)
- string title
- int priority (1–5, where 5 is most urgent)
- string dueDate (format: YYYY-MM-DD)

Required Linked List Operations

Implement a singly linked list with the following functions (you may add more helper functions if needed):

1. **Insert (sorted by priority, then by taskID)**
    - New tasks must be inserted into the correct position so that the list is always sorted:
        - Higher priority comes first
        - If priorities are the same, smaller taskID comes first
        - Reject duplicate taskID
        - Record the operation in history

2. **Search**
    - Search a task by taskID
    - Display the full record if found, otherwise display “Not found”.
    - Record the search operation in history (found / not found)

3. **Delete**
    - Delete a task by taskID
    - Handle all cases correctly (delete head, middle, tail, task not found)
    - Record deletion in history

4. **Display**
    - Display the first **N tasks** (e.g., first 20) in the current linked list
    - Also show the total number of tasks in the list

5. **Statistics**
    - Count how many tasks exist for each priority level (1 to 5)

6. **History**
    - Every insert, delete, or search operation must be logged
    - Each record includes:
        - Action type
        - Task ID

        - Timestamp
    - Display all history records in order
    - Save history records into history.txt

File Processing Requirement

You must support reading and writing task records using a text file:

Input Files

- tasks.txt – initial task records (refer code at below)

Output Files

- tasks_sorted.txt – sorted task list
- history.txt – operation history log

Rules

- History must be recorded **during file loading as well**

Important: Your linked list must be the data structure used for sorting (do not sort using arrays/vectors).

Tasks generation code:

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main() {
    ofstream file("tasks.txt");

    if (!file) {
        cout << "Error creating tasks.txt file.\n";
        return 1;
    }

    srand(time(NULL));

    const int TOTAL_TASKS = 20;

    string titles[TOTAL_TASKS] = {
        "Complete Data Structures Assignment",
        "Prepare Project Proposal",
        "Study for Midterm Exam",
        "Fix Linked List Bugs",
        "Team Meeting Preparation",
        "Update GitHub Repository",
        "Client Feedback Analysis",
        "Database Lab Report",
        "Code Review Session",
        "UI Design Improvement",
        "Test Case Documentation",
        "Submit Research Paper",
        "Cloud Security Presentation",
```

```
        "Finalize Thesis Chapter",
        "Prepare Viva Slides",
        "System Testing Phase",
        "Performance Optimization",
        "Technical Documentation",
        "Project Demonstration",
        "Final Code Submission"
    };

    int baseID = 101;

    for (int i = 0; i < TOTAL_TASKS; i++) {
        int taskID = baseID + i;
        int priority = (rand() % 5) + 1; // 1..5

        int day = (rand() % 28) + 1;
        int month = (rand() % 6) + 1;
        int year = 2025;

        file << taskID << ","
             << titles[i] << ","
             << priority << ","
             << year << "-";

        if (month < 10) file << "0";
        file << month << "-";

        if (day < 10) file << "0";
        file << day << "\n";
    }

    file.close();
    cout << "tasks.txt generated successfully!\n";
    return 0;
}
```

Instruction:

This is a group assignment (2~3). In your report, show sample content of tasks.txt and program screenshot/output after loading by display first 10 tasks. Show also the results of searching 2 taskIDs (1 found, 1 not found); deleting 2 taskIDs (1 head or tail case recommended); display and statistics after deletion; and history log after multiple operations.