COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

NAME:	Hamza Ahmad
REG. NO	SP23-BSE-024
SUBJECT	DS
ASSIGMENT	1
Date	24th September, 2024
SUBMITTED TO	Mr. Muhammad Kamran

Introduction:

The objective of this assignment is to implement a simple task management system using a singly linked list, where each task is represented as a node. The system allows adding tasks based on priority, removing the highest priority task, and removing specific tasks by their ID.

Code Explanation:

TaskNode Struct

- Represents a task in the linked list.
- Contains a unique task ID, a description of the task, a priority level, and a pointer to the next task.

TaskManager Class

- Manages the linked list of tasks.
- Contains a pointer to the head of the list, initialized to `NULL`.

Add Task Function

- Adds a new task based on its priority.
- If the list is empty or the new task has higher priority than the current head, it becomes the new head.
- Otherwise, it finds the correct position to insert the new task, maintaining priority order.

Remove Highest Priority Task

- Removes the task at the head of the list, which has the highest priority.
- If the list is empty, it informs the user.
- Updates the head to the next task and deletes the previous head.

Remove Task by ID

- Removes a specific task using its unique ID.
- If the list is empty, it notifies the user.
- If the task is the head, it updates the head pointer; otherwise, it searches for the task in the list and removes it.

Display Tasks

- Displays all tasks in the list.
- If the list is empty, it informs the user.
- Otherwise, it prints details of each task.

Destructor

- Cleans up memory when a `TaskManager` object is destroyed.
- Continuously removes tasks until the list is empty to prevent memory leaks.

Menu Function

- Provides a console-based interface for user interaction.
- Displays options and calls appropriate functions based on user input.

Main Function

- Entry point of the program.
- Calls the menu function to start the task management system.

CODE:

[*] link_listed_operation.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

struct TaskNode {
    int taskID;
    string description;
    int priority;
    TaskNode* next;
    TaskNode(int id, string desc, int prio) : taskID(id), description(desc), priority(prio), next(NULL) {}
};

class TaskManager {
private:
    TaskNode* head;
public:
    TaskManager() : head(NULL) {}

    // To add a new task
    void addTask(int id, string desc, int prio) {
        TaskNode* newTask = new TaskNode(id, desc, prio);

        if (!head || head->priority < prio) {
            newTask->next = head;
            head = newTask;
            return;
        }
        TaskNode* current = head;
        while (current->next && current->next->priority >= prio) {
            current = current->next;
        }
        newTask->next = current->next;
        current->next = newTask;
    }

    // to display all tasks
    void displayTasks() {
        if (!head) {
            cout << "No tasks available." << endl;
            return;
        }

        TaskNode* current = head;
        cout << "Tasks List:" << endl;
        while (current) {
            cout << "ID: " << current->taskID << ", Description: " << current->description
                 << ", Priority: " << current->priority << endl;
            current = current->next;
        }
    }

    // To remove highest priority task
    void removeHighestPriorityTask() {
        if (!head) {
            cout << "No tasks to remove." << endl;
            return;
        }

        TaskNode* temp = head;
        head = head->next;
        delete temp;
        cout << "Removed highest priority task." << endl;
    }

    // To remove a task by ID
    void removeTaskByID(int id) {
        if (!head) {
            cout << "No tasks to remove." << endl;
            return;
        }
        // task to remove head
        if (head->taskID == id) {
            TaskNode* temp = head;
            head = head->next;
            delete temp;
            cout << "Removed task with ID: " << id << endl;
            return;
        }
        // to find the task with the specified ID
        TaskNode* current = head;
        while (current->next && current->next->taskID != id) {
            current = current->next;
        }
        // If the task was found
        if (current->next) {
            TaskNode* temp = current->next;
            current->next = current->next->next;
            delete temp;
            cout << "Removed task with ID: " << id << endl;
        } else {
            cout << "Task with ID: " << id << " not found." << endl;
```

```cpp
        }
    }

    // free memory allocated for the tasks
    ~TaskManager() {
        while (head) {
            removeHighestPriorityTask();
        }
    }
};

void displayMenu() {
    TaskManager taskManager;
    int choice, id, priority;
    string description;

    do {
        cout << "\nTask Management System" << endl;
        cout << "1. Add Task" << endl;
        cout << "2. View Tasks" << endl;
        cout << "3. Remove Highest Priority Task" << endl;
        cout << "4. Remove Task by ID" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter Task ID: ";
                cin >> id;
                cout << "Enter Task Description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter Priority Level: ";
                cin >> priority;
                taskManager.addTask(id, description, priority);
                break;
            case 2:
                taskManager.displayTasks();
                break;
            case 3:
                taskManager.removeHighestPriorityTask();
                break;
            case 4:
                cout << "Enter Task ID to remove: ";
                cin >> id;
                taskManager.removeTaskByID(id);
                break;
            case 5:
                cout << "Exiting..." << endl;
                break;
            default:
                cout << "Invalid choice, please try again." << endl;
        }
    } while (choice != 5);
}

int main() {
    displayMenu();
    return 0;
}
```

OUTPUT:

```
D:\C++\link_listed_operation.exe

Task Management System
1. Add Task
2. View Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
5. Exit
Enter your choice: 1
Enter Task ID: 112
Enter Task Description: no
Enter Priority Level: 1

Task Management System
1. Add Task
2. View Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
5. Exit
Enter your choice: 2
Tasks List:
ID: 112, Description: no, Priority: 1

Task Management System
1. Add Task
2. View Tasks
3. Remove Highest Priority Task
4. Remove Task by ID
5. Exit
Enter your choice:
```

```
link_listed_operation.cpp

Compile Log   Debug   Find Results   Close

--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\C++\link_listed_operation.exe
- Output Size: 1.8370847702026 4 MiB
- Compilation Time: 1.25s
```