2024

The Task Management System

USMAN SHAFIQ

SP23-BSE-012

24 September 2024

Task: Create a simple task management system using a singly linked list where each task is represented as a node in the list.

Instructions:

1. Each task should contain:

- A unique task ID (integer).
- A task description (string).
- A priority level (integer).

2. Implement the following functionalities:

- Add a new task to the list at the correct position based on priority (higher priority
- tasks come first).
- Remove the task with the highest priority (i.e., delete from the start).
- Remove a specific task using its task ID.

3. Create a console-based menu to:

- Add a new task.
- View all tasks.
- Remove the highest priority task.
- Remove a task by ID.

Introduction

This program implements a simple task management system using a singly linked list. Each task contains a unique ID, description, and priority. The system provides functionalities to add tasks in priority order, remove tasks by priority or ID, and view all tasks. There are five functionalities which are discussed below:

1. **Create Task**
2. **Add Task**
3. **Remove Highest Priority Task**
4. **Remove Task By ID**
5. **View task**

Singly linked list

A data structure known as a single linked list is made up of a series of components known as nodes, each of which consists of two parts:

Data:

- ➢ The actual information or value that is kept on file in the node.

Pointer:

- ➢ A pointer, or reference, to the following node in the series.
  The head, a unique node that points to the list's initial node, is where the list begins. The list's end is indicated by the last node, which points to nullptr.

Explanation of tasks/code:

1. **Structure:**

Three important kinds of data are present in a node:

- **Task ID:** An integer that serves as the task's unique identification.
- **Task Description:** A text that offers specifics about the assignment.
- **Priority:** An integer value indicating the task's importance (greater values correspond to higher priority).
- To create the linked structure, every task node additionally has a reference to the node after it in the list.

2. **Adding a Task Based on Priority**

When a new task is added, it must be inserted into the list in a way that maintains the priority order. The list is traversed to find the correct position for the new task:

- If the list is empty, the new task becomes the “head” (start) of the list.
- If the new task has a higher priority than the current first task, it is inserted at the “beginning” of the list, becoming the new head.
- If the task has a lower priority, the program traverses the list to find the appropriate position. The traversal continues until the new task’s priority is higher than the task at the current position, and it is inserted in the correct spot to maintain the priority order.

3. **Examining Each Task**

The list is scanned from head to tail in order to show every task in the system. For every list node:

- The task is printed along with its ID, description, and priority.
- Once there are no more jobs to display, the traversal continues until the end of the list.

4. **Taking Out the Highest Priority Task**

When a task system is prioritized, the task with the highest priority is always at the top of the list. To make this task disappear:

- Efficiently "skipping over" and eliminating the first task, the head pointer is modified to point to the next node in the list.
- Memory leaks are avoided by freeing up the memory allotted to the deleted task. The highest-priority item is always at the top of the list, making it simple to eliminate, which makes this procedure efficient.

5. Removing a Task by ID

To remove a specific task by its unique ID, the list is traversed from the head:

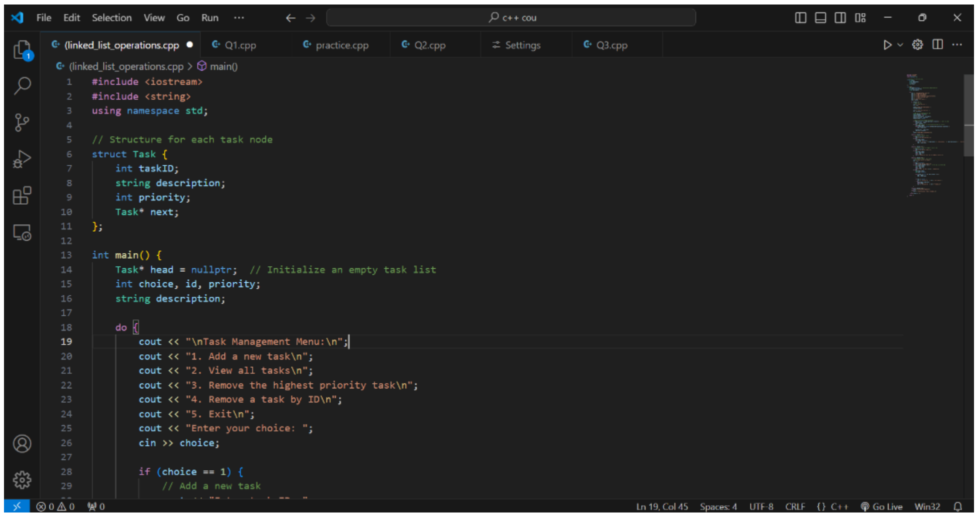
- The program checks each node's task ID until it finds the one that matches the user’s input.
- If the task is found at the head, it is removed just like the highest-priority task.
- If the task is found elsewhere in the list, the program adjusts the pointers of the surrounding nodes to "skip over" the task, effectively removing it.

- If the task with the given ID is not found, an appropriate message is displayed to indicate that the task does not exist.

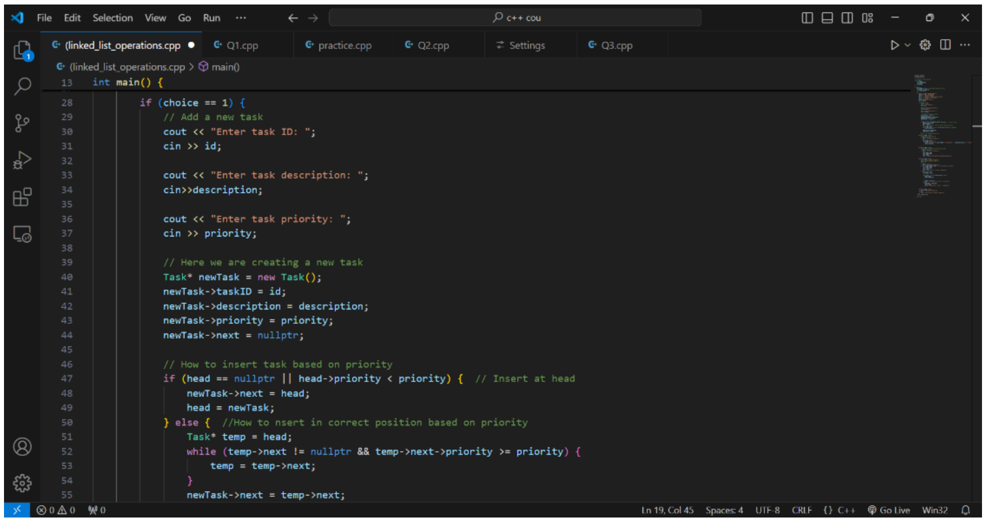
Conclusion:

My understanding of linked list management for a practical task management system has improved because of this project. I gained knowledge about managing tasks generally, handling insertion according to priority, and deleting nodes. Handling edge cases and pointers was difficult but useful.

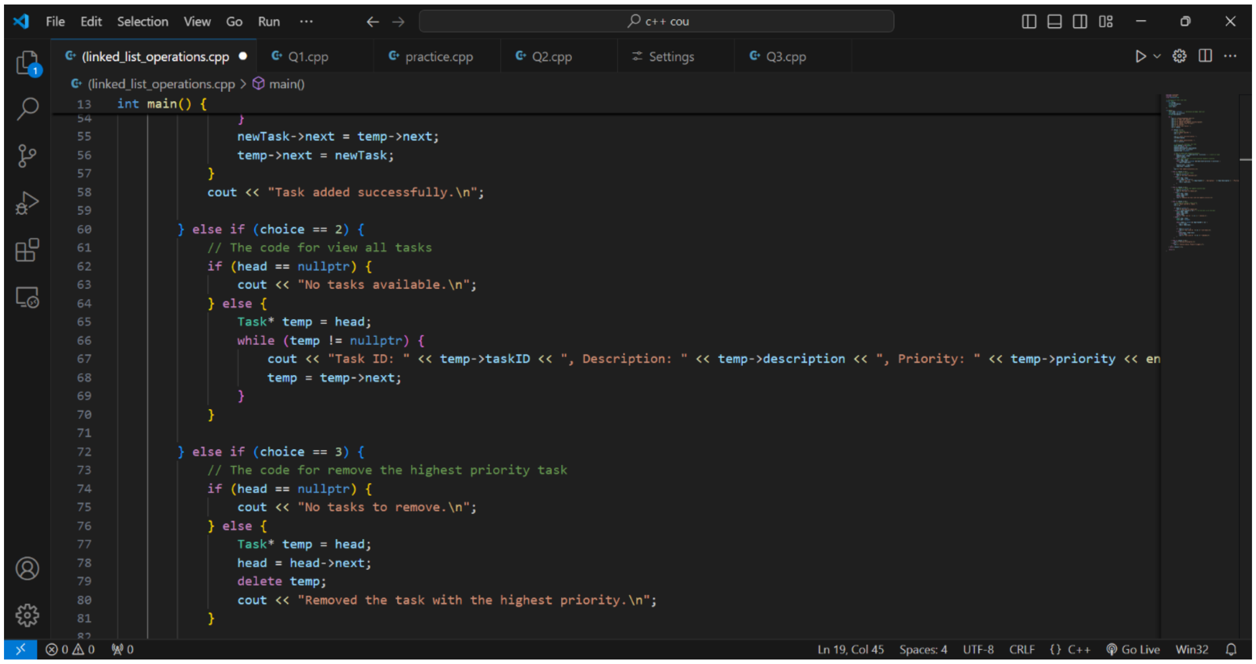
Screen Shots Of Code And Outputs In VS Studio



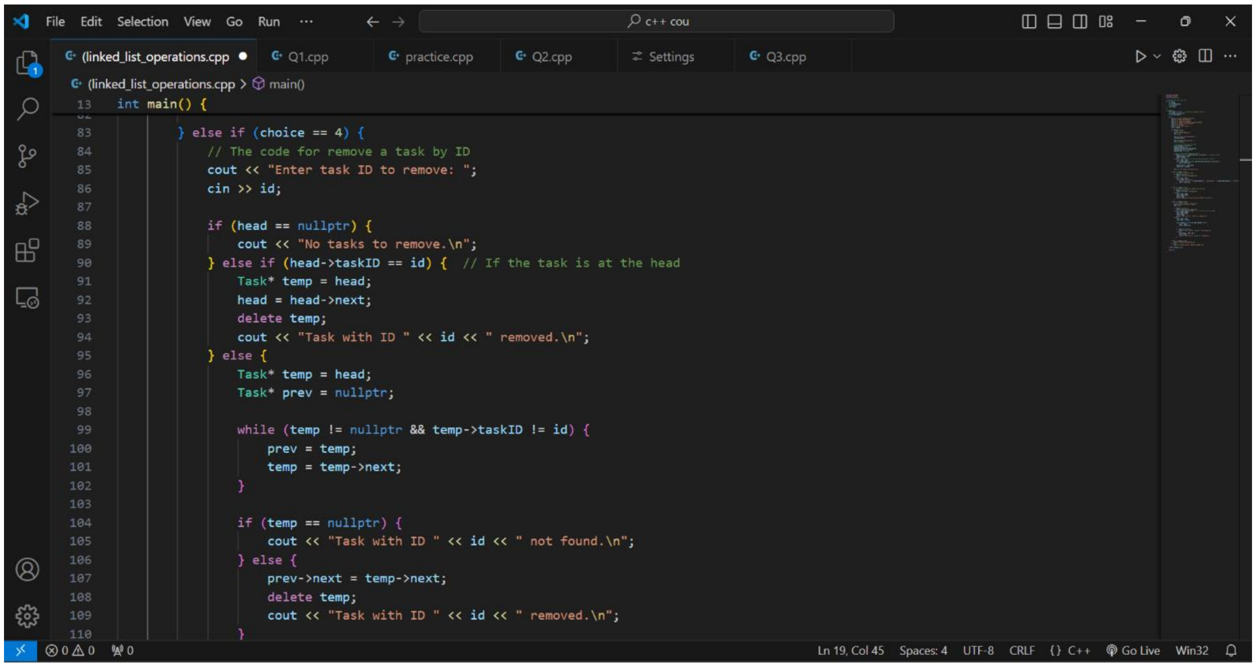
```cpp
#include <iostream>
#include <string>
using namespace std;

// Structure for each task node
struct Task {
    int taskID;
    string description;
    int priority;
    Task* next;
};

int main() {
    Task* head = nullptr;  // Initialize an empty task list
    int choice, id, priority;
    string description;

    do {
        cout << "\nTask Management Menu:\n";
        cout << "1. Add a new task\n";
        cout << "2. View all tasks\n";
        cout << "3. Remove the highest priority task\n";
        cout << "4. Remove a task by ID\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 1) {
            // Add a new task
```



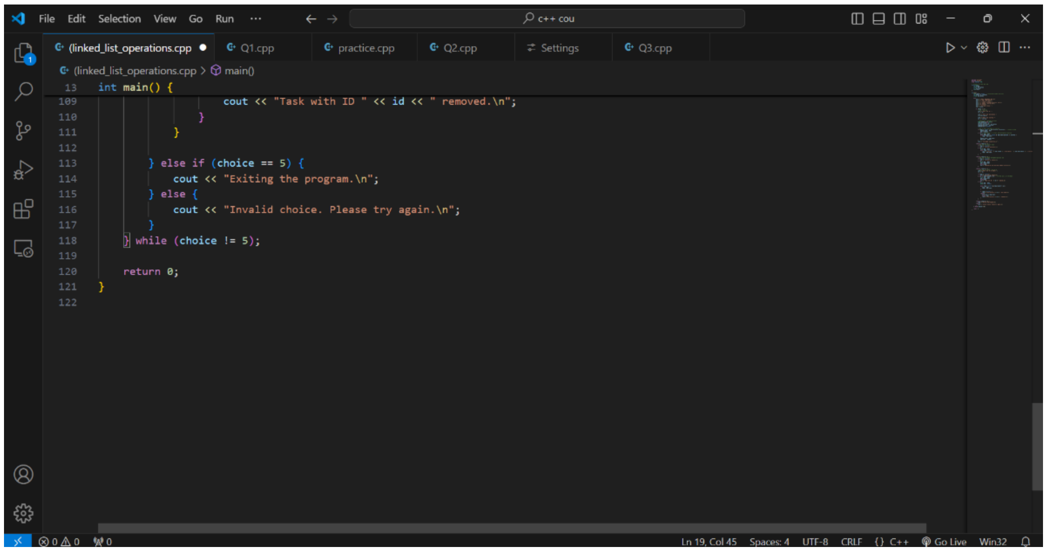
```cpp
int main() {
        if (choice == 1) {
            // Add a new task
            cout << "Enter task ID: ";
            cin >> id;

            cout << "Enter task description: ";
            cin>>description;

            cout << "Enter task priority: ";
            cin >> priority;

            // Here we are creating a new task
            Task* newTask = new Task();
            newTask->taskID = id;
            newTask->description = description;
            newTask->priority = priority;
            newTask->next = nullptr;

            // How to insert task based on priority
            if (head == nullptr || head->priority < priority) {  // Insert at head
                newTask->next = head;
                head = newTask;
            } else {  //How to nsert in correct position based on priority
                Task* temp = head;
                while (temp->next != nullptr && temp->next->priority >= priority) {
                    temp = temp->next;
                }
                newTask->next = temp->next;
```


File Edit Selection View Go Run ···
c++ cou
(linked_list_operations.cpp
Q1.cpp
practice.cpp
Q2.cpp
Settings
Q3.cpp
(linked_list_operations.cpp > main()
int main() {
Ln 19, Col 45 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32

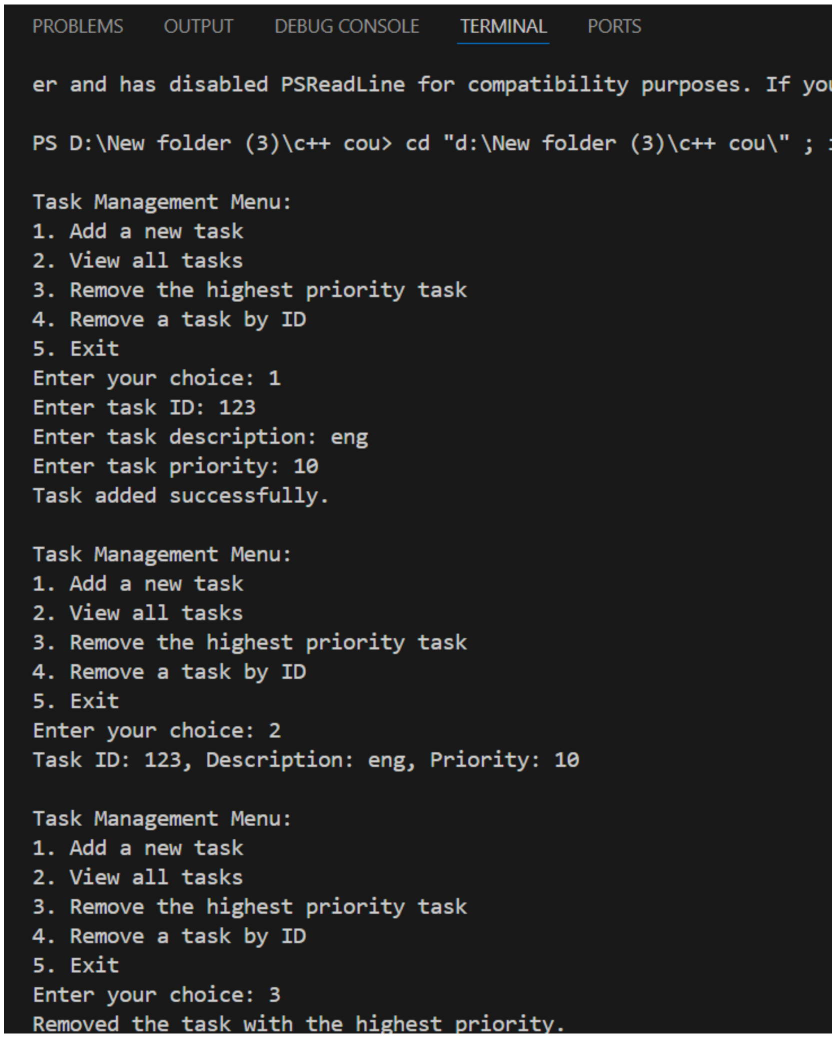
```
                }
                newTask->next = temp->next;
                temp->next = newTask;
            }
            cout << "Task added successfully.\n";

        } else if (choice == 2) {
            // The code for view all tasks
            if (head == nullptr) {
                cout << "No tasks available.\n";
            } else {
                Task* temp = head;
                while (temp != nullptr) {
                    cout << "Task ID: " << temp->taskID << ", Description: " << temp->description << ", Priority: " << temp->priority << en
                    temp = temp->next;
                }
            }

        } else if (choice == 3) {
            // The code for remove the highest priority task
            if (head == nullptr) {
                cout << "No tasks to remove.\n";
            } else {
                Task* temp = head;
                head = head->next;
                delete temp;
                cout << "Removed the task with the highest priority.\n";
            }
```


File Edit Selection View Go Run ···
c++ cou
(linked_list_operations.cpp
Q1.cpp
practice.cpp
Q2.cpp
Settings
Q3.cpp
(linked_list_operations.cpp > main()
int main() {
Ln 19, Col 45 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32

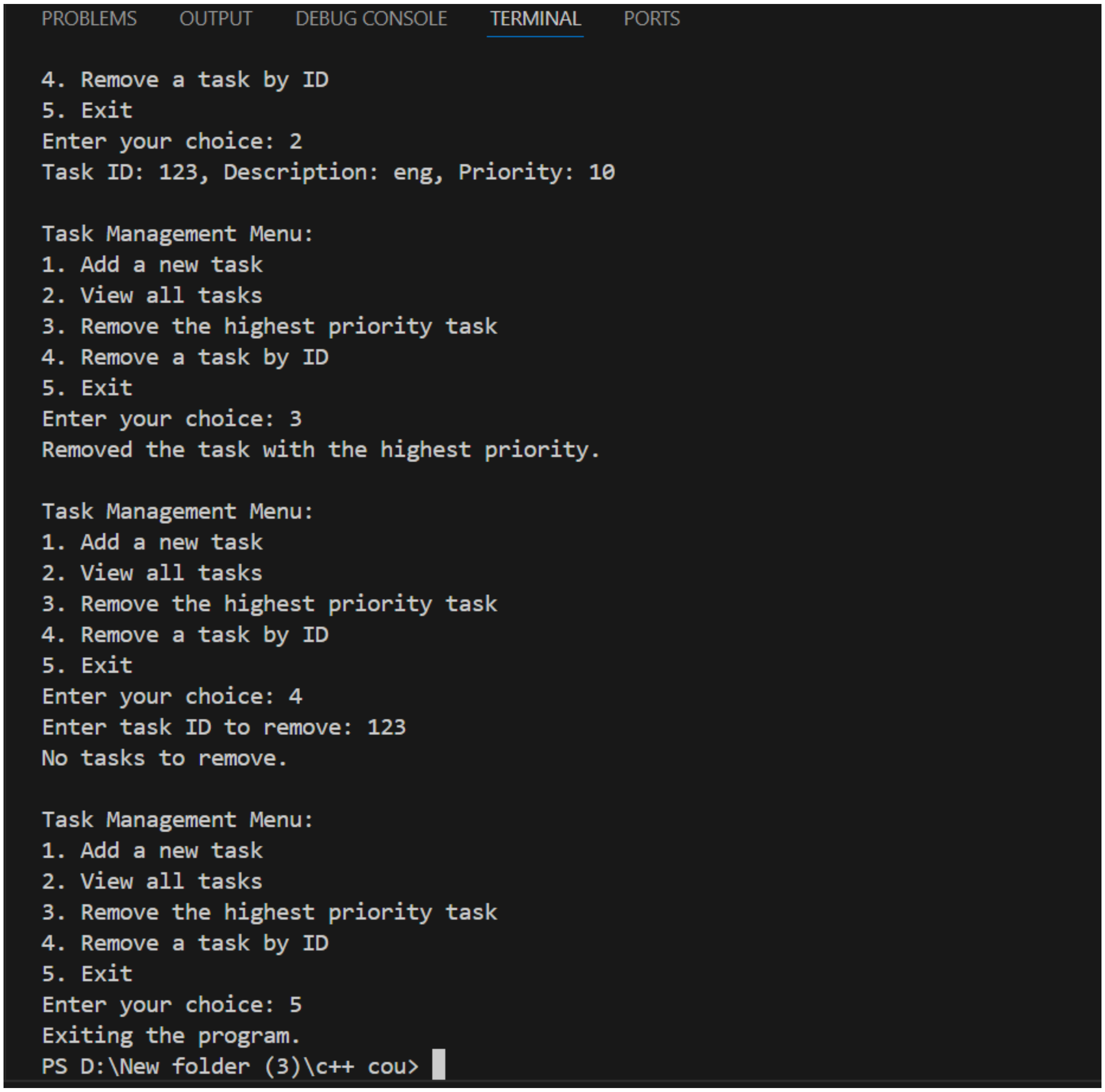
```
        } else if (choice == 4) {
            // The code for remove a task by ID
            cout << "Enter task ID to remove: ";
            cin >> id;

            if (head == nullptr) {
                cout << "No tasks to remove.\n";
            } else if (head->taskID == id) {  // If the task is at the head
                Task* temp = head;
                head = head->next;
                delete temp;
                cout << "Task with ID " << id << " removed.\n";
            } else {
                Task* temp = head;
                Task* prev = nullptr;

                while (temp != nullptr && temp->taskID != id) {
                    prev = temp;
                    temp = temp->next;
                }

                if (temp == nullptr) {
                    cout << "Task with ID " << id << " not found.\n";
                } else {
                    prev->next = temp->next;
                    delete temp;
                    cout << "Task with ID " << id << " removed.\n";
                }
```


File Edit Selection View Go Run
c++ cou
(linked_list_operations.cpp
Q1.cpp
practice.cpp
Q2.cpp
Settings
Q3.cpp
(linked_list_operations.cpp > main()
Ln 19, Col 45 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32

```cpp
int main() {
                    cout << "Task with ID " << id << " removed.\n";
                }
            }

        } else if (choice == 5) {
            cout << "Exiting the program.\n";
        } else {
            cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 5);

    return 0;
}
```



```
er and has disabled PSReadLine for compatibility purposes. If yo

PS D:\New folder (3)\c++ cou> cd "d:\New folder (3)\c++ cou\" ;

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 1
Enter task ID: 123
Enter task description: eng
Enter task priority: 10
Task added successfully.

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 2
Task ID: 123, Description: eng, Priority: 10

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 3
Removed the task with the highest priority.
```



```
4. Remove a task by ID
5. Exit
Enter your choice: 2
Task ID: 123, Description: eng, Priority: 10

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 3
Removed the task with the highest priority.

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 4
Enter task ID to remove: 123
No tasks to remove.

Task Management Menu:
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 5
Exiting the program.
PS D:\New folder (3)\c++ cou>
```