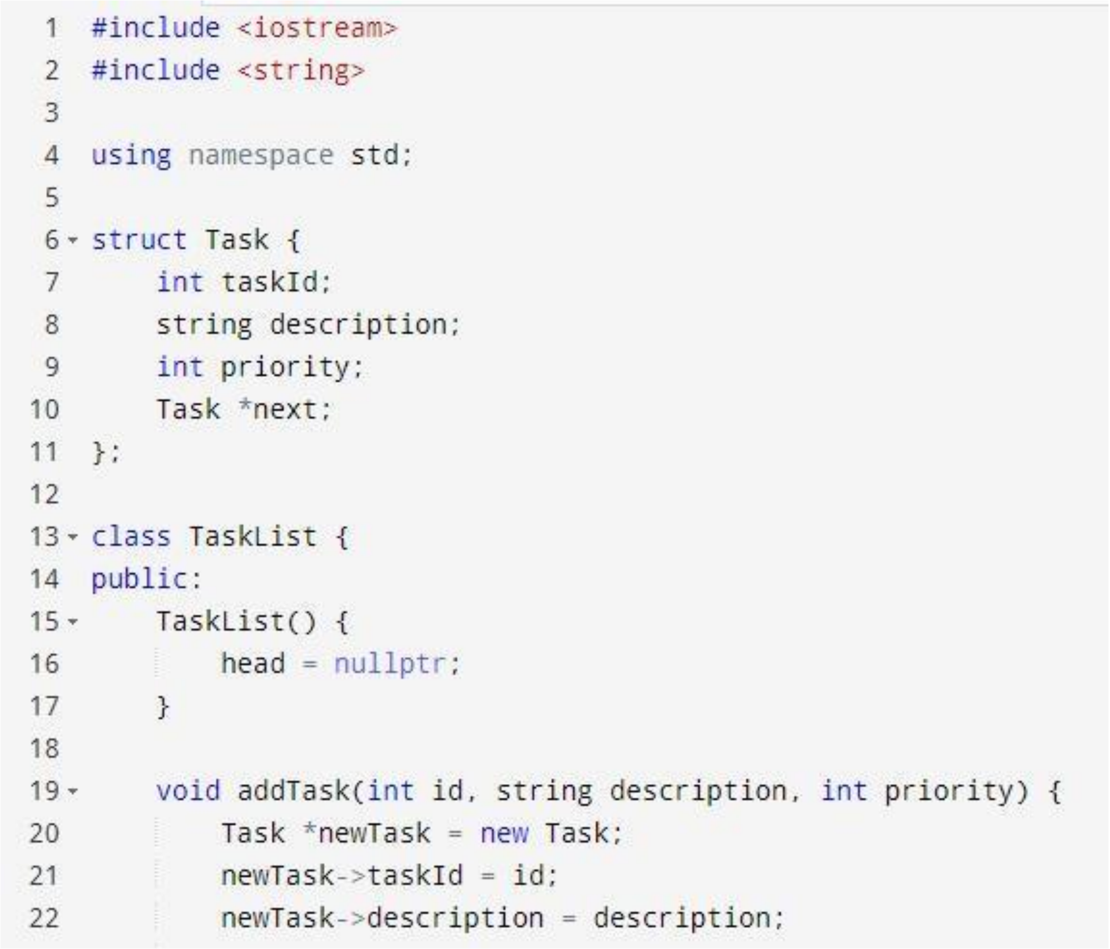
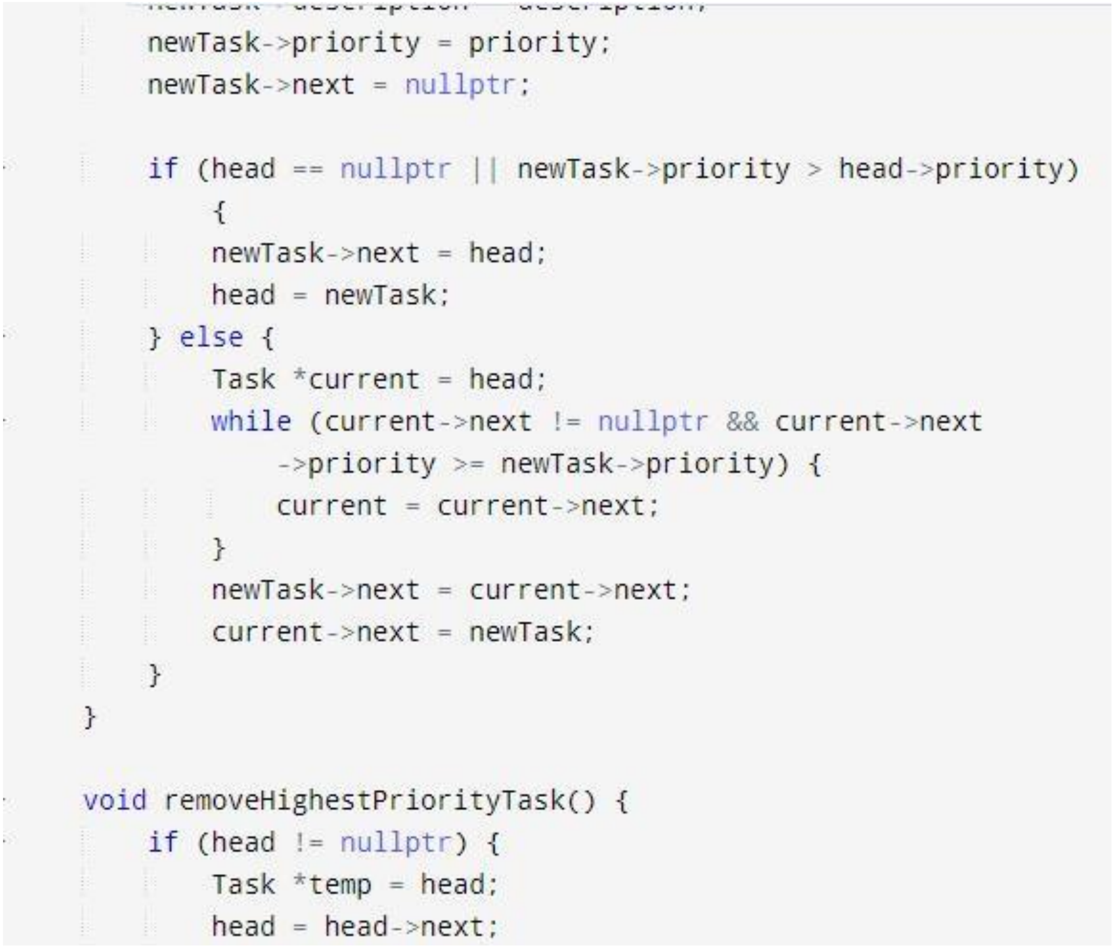
COMSATS UNIVERSITY ISLAMABAD
ATTOCK CAMPUS

DEPARTMENT OF COMPUTER SCIENCE

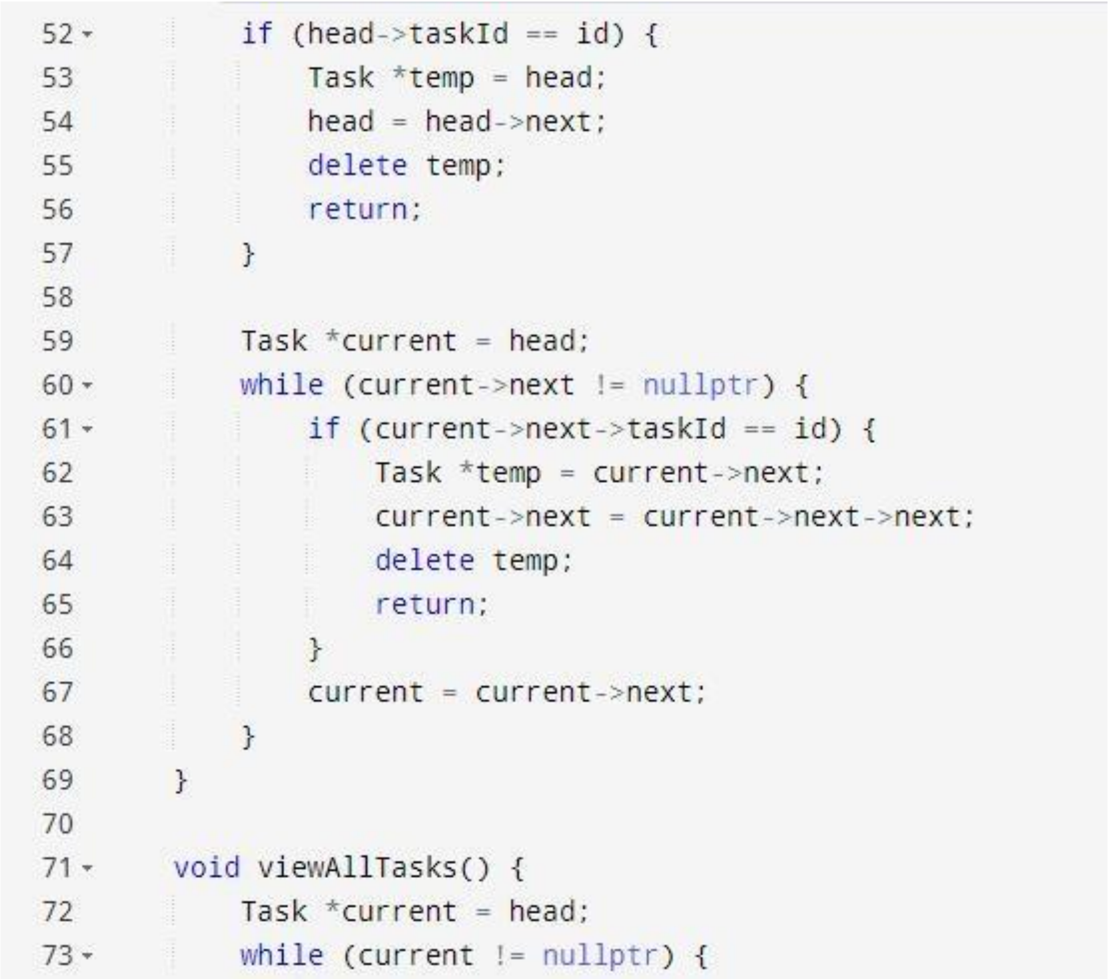
NAME	***:***	***Arsalan Ahmed***
REG. NO	***:***	***FA22-BSE-039***
SUBJECT	***:***	***Data Structures***
ASSIGNMENT	***:***	***01***
Date	***:***	***24 September, 2024***
SUBMITTED TO	***:***	***Mr Kamran***



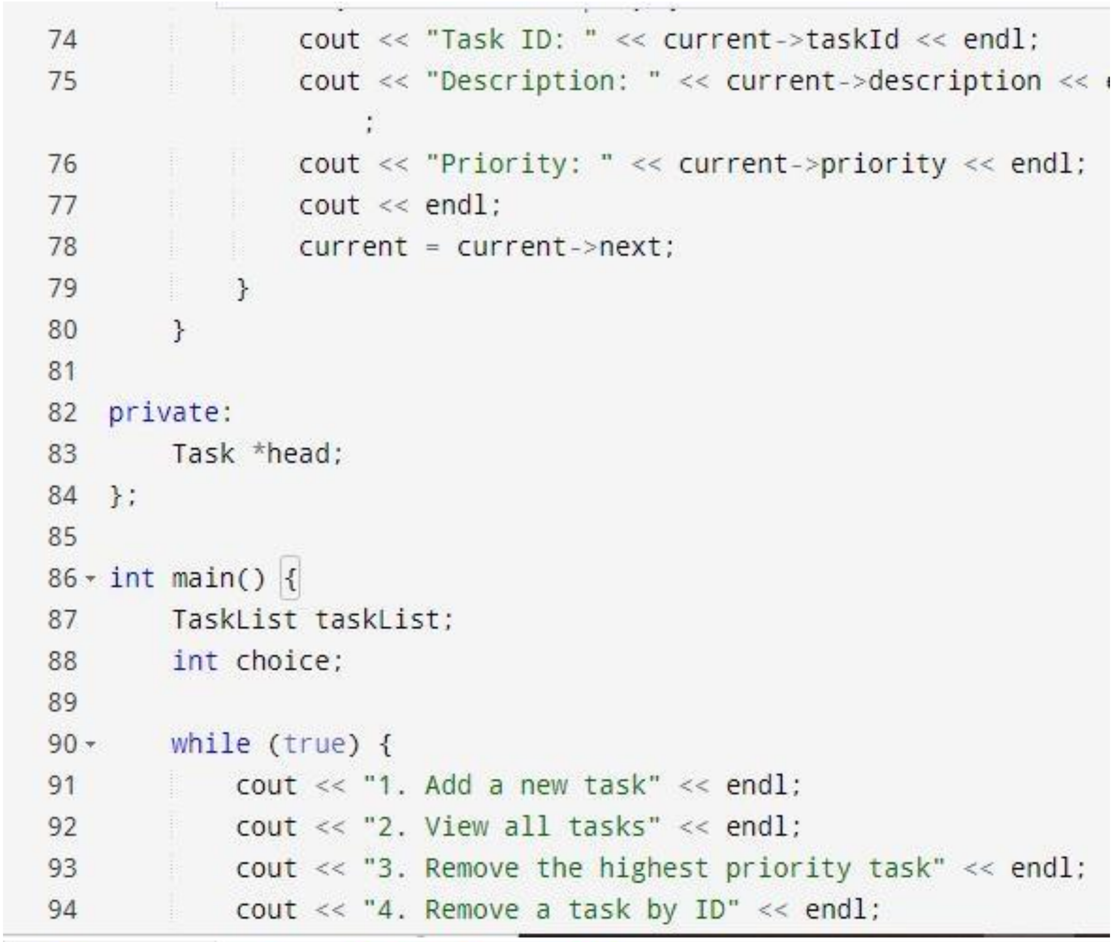
```cpp
#include <iostream>
#include <string>

using namespace std;

struct Task {
    int taskId;
    string description;
    int priority;
    Task *next;
};

class TaskList {
public:
    TaskList() {
        head = nullptr;
    }

    void addTask(int id, string description, int priority) {
        Task *newTask = new Task;
        newTask->taskId = id;
        newTask->description = description;
```



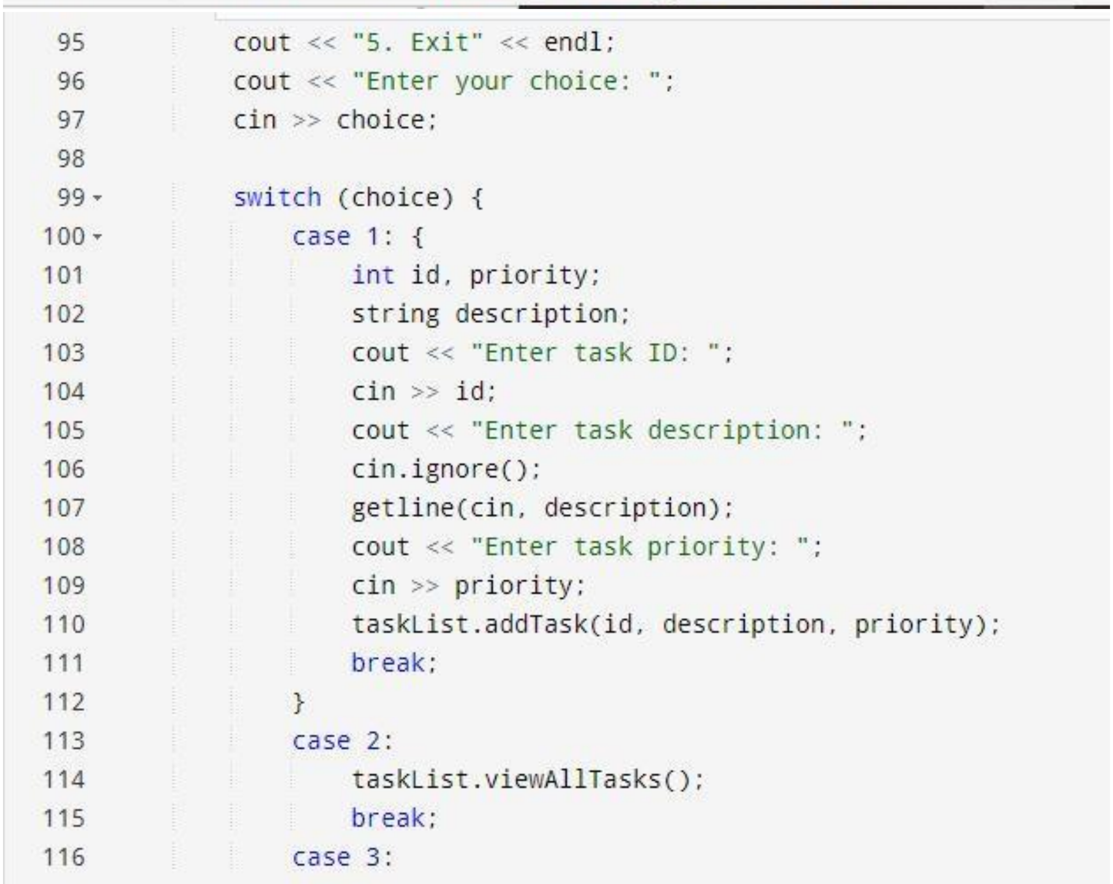
```
    newTask->description = description;
    newTask->priority = priority;
    newTask->next = nullptr;

    if (head == nullptr || newTask->priority > head->priority)
        {
        newTask->next = head;
        head = newTask;
    } else {
        Task *current = head;
        while (current->next != nullptr && current->next
            ->priority >= newTask->priority) {
            current = current->next;
        }
        newTask->next = current->next;
        current->next = newTask;
    }
}

void removeHighestPriorityTask() {
    if (head != nullptr) {
        Task *temp = head;
        head = head->next;
```



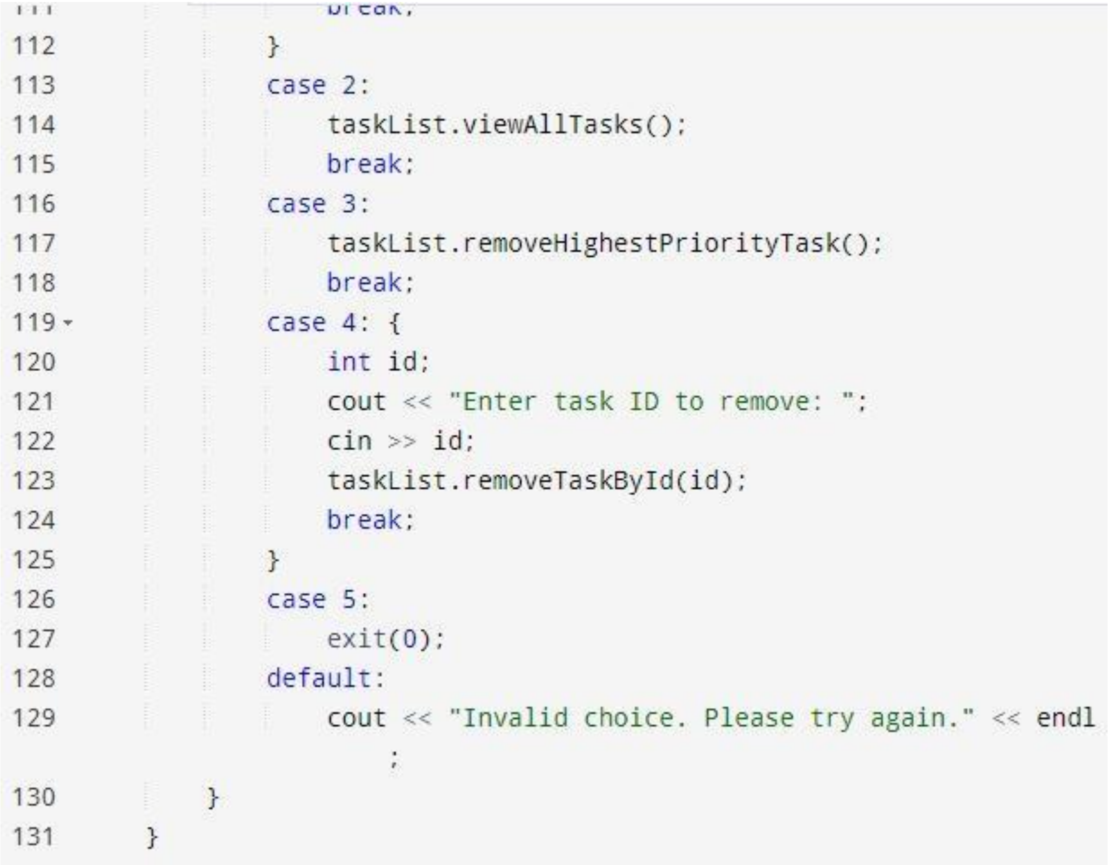
```
    if (head->taskId == id) {
        Task *temp = head;
        head = head->next;
        delete temp;
        return;
    }

    Task *current = head;
    while (current->next != nullptr) {
        if (current->next->taskId == id) {
            Task *temp = current->next;
            current->next = current->next->next;
            delete temp;
            return;
        }
        current = current->next;
    }
}

void viewAllTasks() {
    Task *current = head;
    while (current != nullptr) {
```



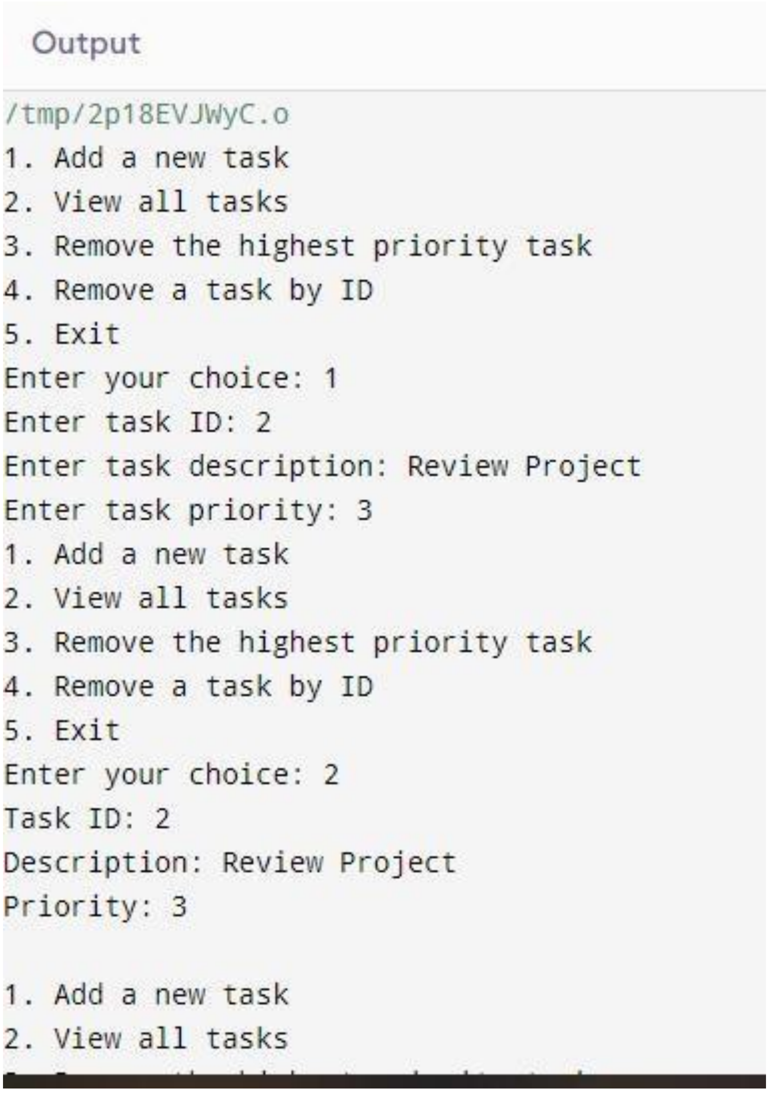
```
            cout << "Task ID: " << current->taskId << endl;
            cout << "Description: " << current->description << 
                ;
            cout << "Priority: " << current->priority << endl;
            cout << endl;
            current = current->next;
        }
    }

private:
    Task *head;
};

int main() {
    TaskList taskList;
    int choice;

    while (true) {
        cout << "1. Add a new task" << endl;
        cout << "2. View all tasks" << endl;
        cout << "3. Remove the highest priority task" << endl;
        cout << "4. Remove a task by ID" << endl;
```



```
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int id, priority;
                string description;
                cout << "Enter task ID: ";
                cin >> id;
                cout << "Enter task description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter task priority: ";
                cin >> priority;
                taskList.addTask(id, description, priority);
                break;
            }
            case 2:
                taskList.viewAllTasks();
                break;
            case 3:
```



```
                break;
            }
            case 2:
                taskList.viewAllTasks();
                break;
            case 3:
                taskList.removeHighestPriorityTask();
                break;
            case 4: {
                int id;
                cout << "Enter task ID to remove: ";
                cin >> id;
                taskList.removeTaskById(id);
                break;
            }
            case 5:
                exit(0);
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }
```



Output

```
/tmp/2p18EVJWyC.o
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: 2
Enter task description: Review Project
Enter task priority: 3
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: 2
Description: Review Project
Priority: 3

1. Add a new task
2. View all tasks
```

Report

Introduction

This assignment aims to create a simple task management system using object-oriented programming (OOP) in C++. We will use a linked list to manage tasks, where each task has an ID, a description, and a priority. Users will be able to add new tasks, remove tasks based on ID or priority, and view all tasks sorted by priority. The main goals are to understand dynamic memory management, build linked lists, and learn basic task management operations.

Code Interpretation

Task Structure

Each task has:

- **Tasked**: A unique number identifying the task.
- **Description**: A brief text about the task.
- **Priority**: An integer indicating how important the task is; higher numbers mean higher priority.
- **Next**: A pointer that links to the next task, forming a linked list.

Task List Class

The `Task List` class manages all tasks. It has functions for adding, removing, and viewing tasks.

Builder

The constructor initializes the task list, setting the head pointer to `nullptr`. This pointer marks the start of the linked list.

Destructor

The destructor ensures that all memory used by the tasks is freed when the `TaskList` object is destroyed. It goes through the list and deletes each task to prevent memory leaks.

Add Task(int priority, int id, string description)

This function adds a new task to the list in the correct position based on its priority. If the list is empty or the new task is more important than the current head task, it becomes the new head. Otherwise, the function finds the right spot for the new task by moving through the list.

Delete Task with Maximum Priority

This function removes the task with the highest priority, which is always the first task in the list (the head). If there are no tasks, it informs the user.

Remove Task by ID (int id)

This function removes a task based on its ID. If the task is the head, it deletes the head. If not, it searches the list for the task with the given ID. If no task is found, it informs the user.

View All Tasks

This function starts from the head of the list and prints each task's ID, description, and priority. If there are no tasks, it informs the user that the list is empty.

Main Function

In the `main()` function, there is a loop that presents a menu with options to add a task, view all tasks, remove the highest priority task, remove a task by ID, or exit the program. The user can select an option, and the corresponding method from the `Task List` class is called. The loop continues until the user chooses to exit.

Input validation is included to ensure the user enters valid data.

Conclusion

Through this assignment, I learned how to use dynamic data structures like linked lists in C++. I reinforced key OOP concepts, including data abstraction, constructors, destructors, and pointers. One of the biggest challenges was ensuring that all operations, especially those involving memory allocation and task management, were performed correctly.

I also gained experience with error handling and input validation, which are essential for creating reliable programs. Finally, I learned the importance of properly managing memory to avoid leaks and bugs, which can be tricky to fix in more complex programs.