COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS

CS DEPARTMENT

NAME	**:**	**JANEETA ISHTIAQ**
REG NO	**:**	**SP23-BSE-040**
COURSE	**:**	**DS (THEORY)**
ASSIGNMENT NO	**:**	**01**
DATE	**:**	**24 September 2024**
SUBMITTED TO	**:**	**SIR KAMRAN**

Introduction:

The objective of this assignment is to create a basic task management system using a priority-based linked list. This program allows users to add tasks, view all tasks, remove the highest priority task, and remove a task by its ID. The key operations are implemented using fundamental concepts like dynamic memory allocation, structures, and linked list manipulation. Tasks are prioritized using an integer value where a higher priority task is inserted before lower priority ones.

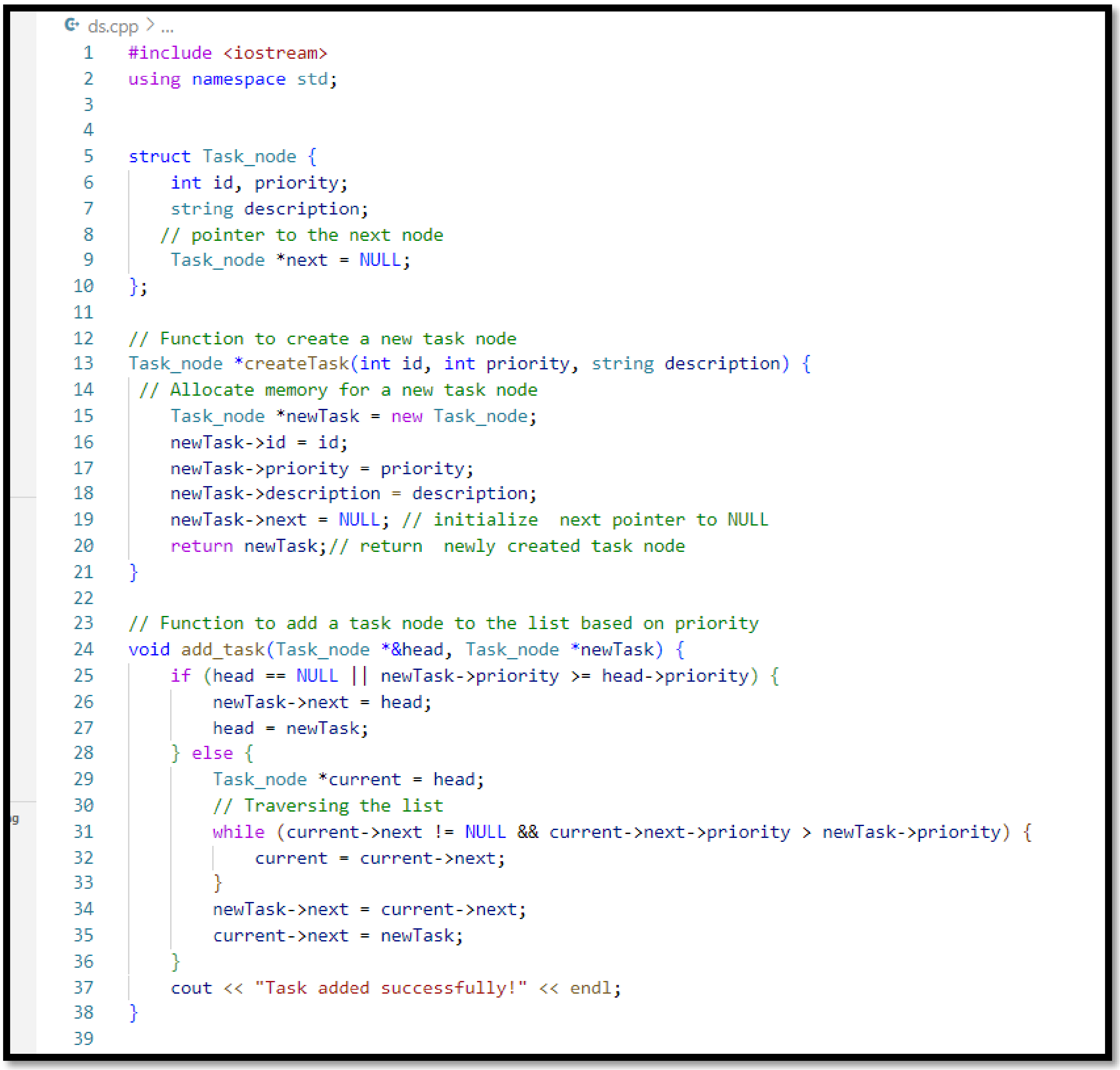
Code Explanation:

1. **struct Task_node:**
    - A structure that defines the properties of each task. It includes:
        - id: A unique identifier for each task.
        - priority: An integer value that defines the task's importance. Higher numbers mean higher priority.
        - description: A string that describes the task.
        - next: A pointer that points to the next task node in the linked list.
2. **createTask(int id, int priority, string description):**
    - This function dynamically allocates memory for a new task node and initializes its id, priority, and description attributes. It also sets the next pointer to NULL.
3. **add_task(Task_node *&head, Task_node *newTask):**
    - This function adds a new task to the linked list in descending order of priority. If the task list is empty or the new task has a higher priority than the head, it becomes the new head. Otherwise, it traverses the list and inserts the task at the appropriate position based on its priority.
4. **remove_on_priority(Task_node *&head):**
    - This function removes the task with the highest priority, i.e., the head of the list. The memory allocated to the removed task is released using delete, and the head is updated to the next node.
5. **remove_by_id(Task_node *&head, int id):**
    - This function removes a task by its id. It first checks if the head contains the task with the given id. If not, it traverses the list, searching for the task, and removes it once found.
6. **view_tasks(Task_node *head):**
    - This function traverses the entire task list and prints the details of each task, including its ID, description, and priority.
7. **main():**
    - The main function handles the user interface. It runs an infinite loop that displays a menu where the user can choose to:
        1. Add a new task.
        2. View all tasks.
        3. Remove the highest priority task.
        4. Remove a task by ID.

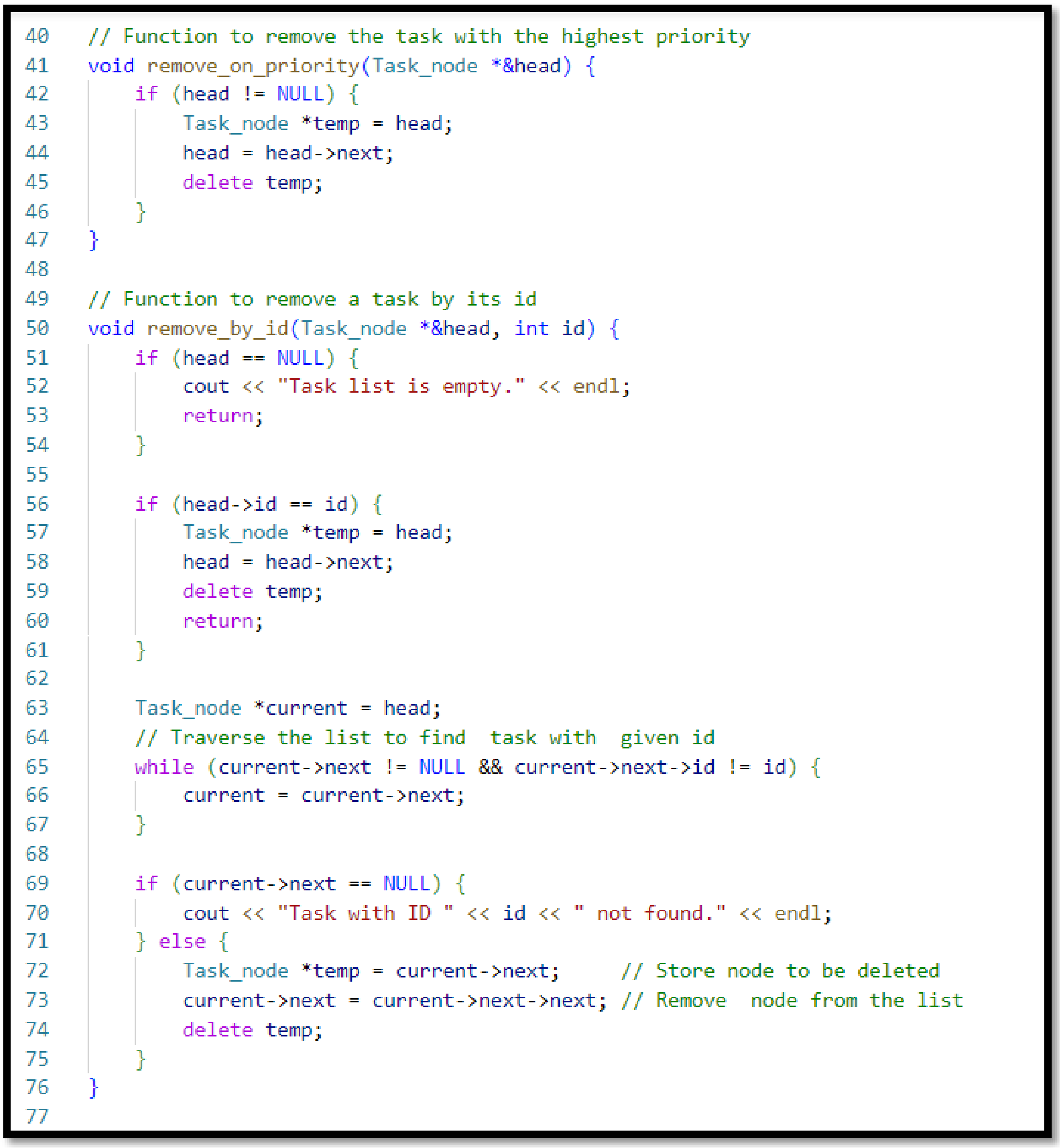
5. Exit the program.

- Based on the user's choice, the corresponding function is called. The program continues until the user chooses to exit.

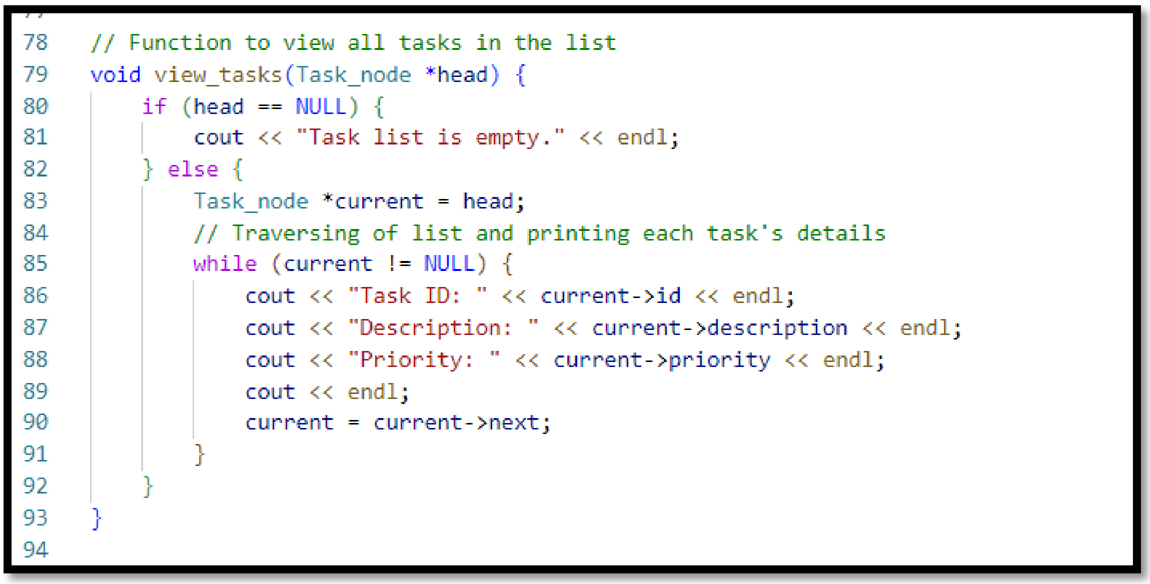
CODE:



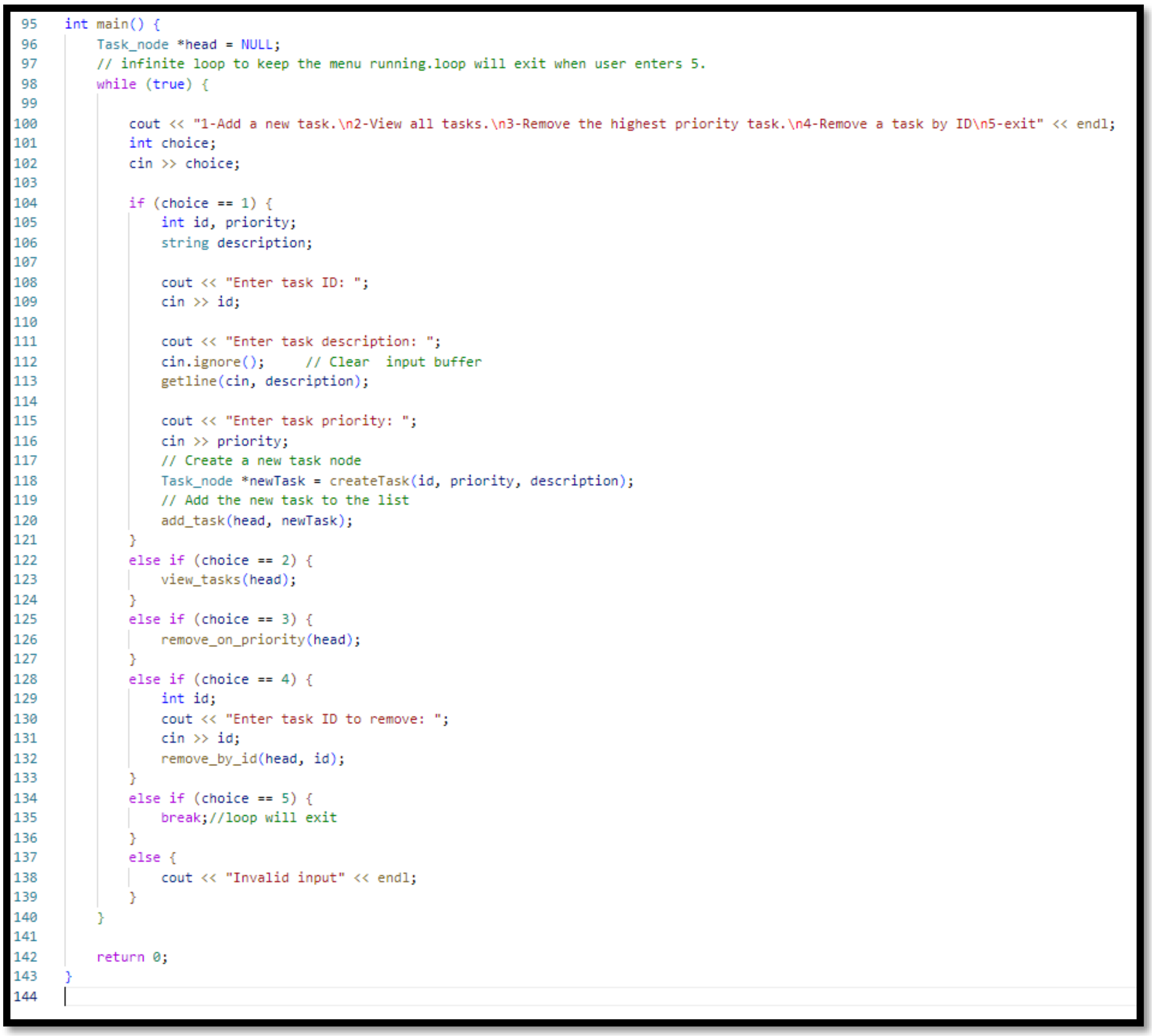
```cpp
#include <iostream>
using namespace std;


struct Task_node {
    int id, priority;
    string description;
   // pointer to the next node
    Task_node *next = NULL;
};

// Function to create a new task node
Task_node *createTask(int id, int priority, string description) {
 // Allocate memory for a new task node
    Task_node *newTask = new Task_node;
    newTask->id = id;
    newTask->priority = priority;
    newTask->description = description;
    newTask->next = NULL; // initialize  next pointer to NULL
    return newTask;// return  newly created task node
}

// Function to add a task node to the list based on priority
void add_task(Task_node *&head, Task_node *newTask) {
    if (head == NULL || newTask->priority >= head->priority) {
        newTask->next = head;
        head = newTask;
    } else {
        Task_node *current = head;
        // Traversing the list
        while (current->next != NULL && current->next->priority > newTask->priority) {
            current = current->next;
        }
        newTask->next = current->next;
        current->next = newTask;
    }
    cout << "Task added successfully!" << endl;
}

```



```cpp
// Function to remove the task with the highest priority
void remove_on_priority(Task_node *&head) {
    if (head != NULL) {
        Task_node *temp = head;
        head = head->next;
        delete temp;
    }
}

// Function to remove a task by its id
void remove_by_id(Task_node *&head, int id) {
    if (head == NULL) {
        cout << "Task list is empty." << endl;
        return;
    }

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

    Task_node *current = head;
    // Traverse the list to find  task with  given id
    while (current->next != NULL && current->next->id != id) {
        current = current->next;
    }

    if (current->next == NULL) {
        cout << "Task with ID " << id << " not found." << endl;
    } else {
        Task_node *temp = current->next;      // Store node to be deleted
        current->next = current->next->next; // Remove  node from the list
        delete temp;
    }
}

```

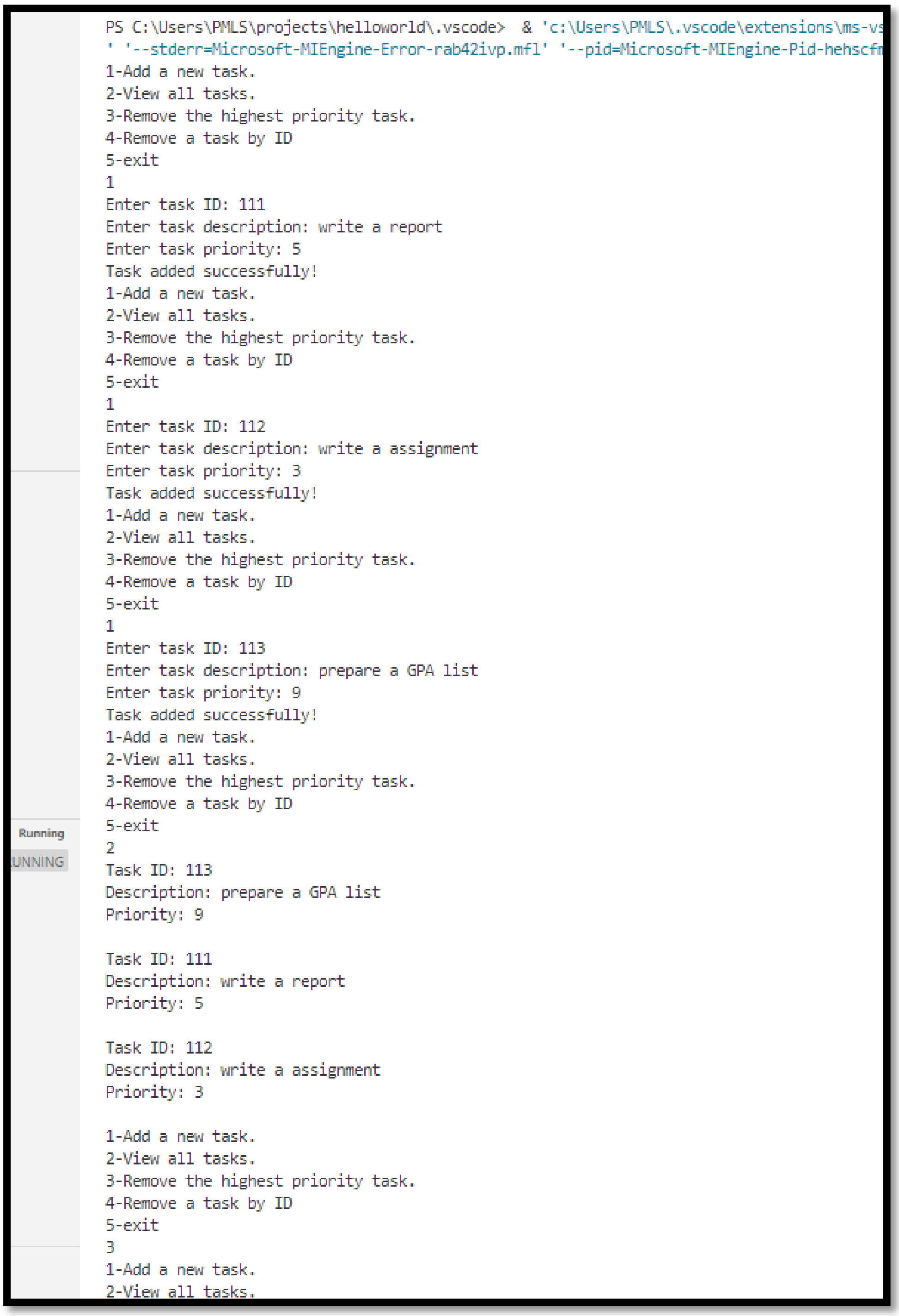


```cpp
// Function to view all tasks in the list
void view_tasks(Task_node *head) {
    if (head == NULL) {
        cout << "Task list is empty." << endl;
    } else {
        Task_node *current = head;
        // Traversing of list and printing each task's details
        while (current != NULL) {
            cout << "Task ID: " << current->id << endl;
            cout << "Description: " << current->description << endl;
            cout << "Priority: " << current->priority << endl;
            cout << endl;
            current = current->next;
        }
    }
}

```

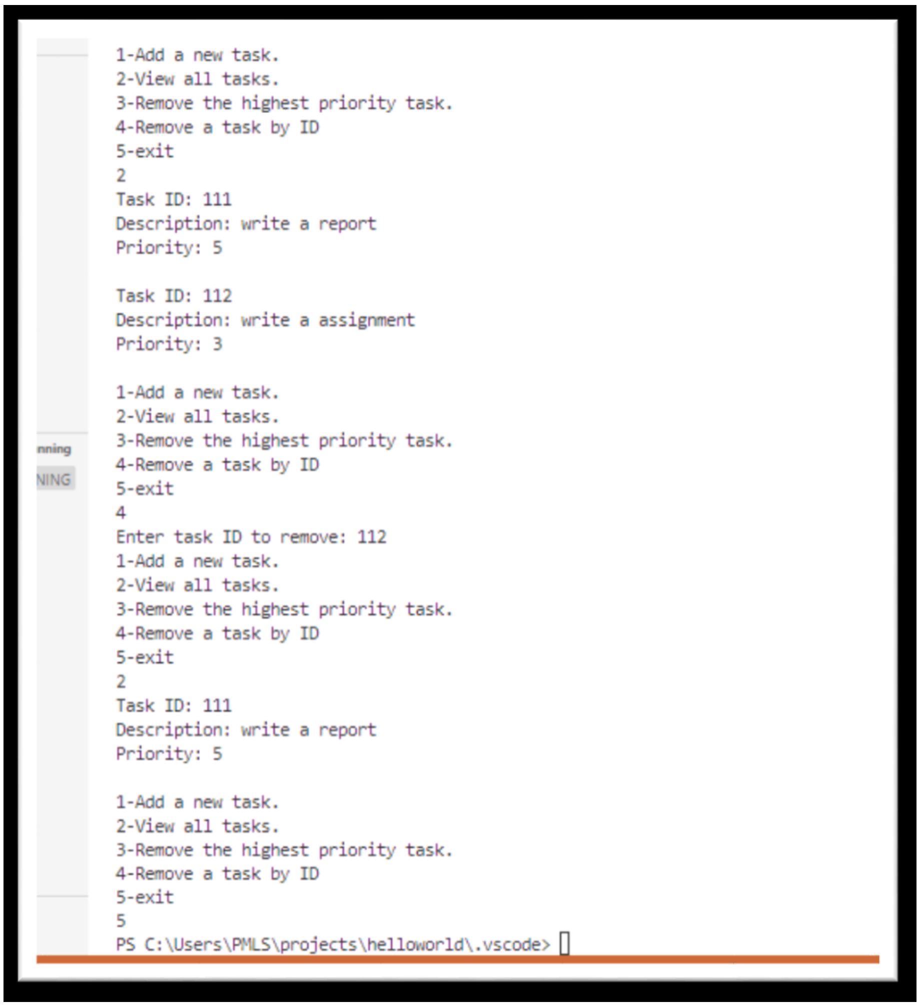


```cpp
int main() {
    Task_node *head = NULL;
    // infinite loop to keep the menu running.loop will exit when user enters 5.
    while (true) {

        cout << "1-Add a new task.\n2-View all tasks.\n3-Remove the highest priority task.\n4-Remove a task by ID\n5-exit" << endl;
        int choice;
        cin >> choice;

        if (choice == 1) {
            int id, priority;
            string description;

            cout << "Enter task ID: ";
            cin >> id;

            cout << "Enter task description: ";
            cin.ignore();     // Clear  input buffer
            getline(cin, description);

            cout << "Enter task priority: ";
            cin >> priority;
            // Create a new task node
            Task_node *newTask = createTask(id, priority, description);
            // Add the new task to the list
            add_task(head, newTask);
        }
        else if (choice == 2) {
            view_tasks(head);
        }
        else if (choice == 3) {
            remove_on_priority(head);
        }
        else if (choice == 4) {
            int id;
            cout << "Enter task ID to remove: ";
            cin >> id;
            remove_by_id(head, id);
        }
        else if (choice == 5) {
            break;//loop will exit
        }
        else {
            cout << "Invalid input" << endl;
        }
    }

    return 0;
}
```

OUTPUT:



```
PS C:\Users\PMLS\projects\helloworld\.vscode>  & 'c:\Users\PMLS\.vscode\extensions\ms-vs
' '--stderr=Microsoft-MIEngine-Error-rab42ivp.mfl' '--pid=Microsoft-MIEngine-Pid-hehscfm
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
1
Enter task ID: 111
Enter task description: write a report
Enter task priority: 5
Task added successfully!
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
1
Enter task ID: 112
Enter task description: write a assignment
Enter task priority: 3
Task added successfully!
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
1
Enter task ID: 113
Enter task description: prepare a GPA list
Enter task priority: 9
Task added successfully!
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
2
Task ID: 113
Description: prepare a GPA list
Priority: 9

Task ID: 111
Description: write a report
Priority: 5

Task ID: 112
Description: write a assignment
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
3
1-Add a new task.
2-View all tasks.
```



```
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
2
Task ID: 111
Description: write a report
Priority: 5

Task ID: 112
Description: write a assignment
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
4
Enter task ID to remove: 112
1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
2
Task ID: 111
Description: write a report
Priority: 5

1-Add a new task.
2-View all tasks.
3-Remove the highest priority task.
4-Remove a task by ID
5-exit
5
PS C:\Users\PMLS\projects\helloworld\.vscode> 
```

Conclusion:

Through this assignment, I learned how to manage dynamic memory using pointers and implement linked lists for task management based on priority. Understanding how to manipulate linked lists, handle dynamic memory with new and delete.

One challenge I faced was ensuring that tasks are correctly inserted into the list in order of priority, especially when the list is empty or when the new task has the highest priority..