COMSATS University Islamabad

Attock campus

Program: BSE

Name:	**Qurat-ul-Ain Bibi**
Registration #:	**SP23-BSE-032**
Course:	**DS-Theory**
Date:	**24-09-2024**
Assignment #:	**01**
Submitted To:	**Sir Muhammad Kamran**

Introduction:

The objective of this assignment is to design and implement a task management system using a singly linked list. The system allows users to add new tasks, view all tasks, remove the highest priority task, and remove a task by ID.

Code Explanation:

1. addTask()

Adds a new task to the linked list, maintaining priority order. Checks for duplicate task IDs and inserts the new task at the correct position based on its priority.

2. viewTasks()

Displays all tasks in the linked list, printing task ID, description, and priority level for each task. Traverses the list from the head node to the end.

3. removeHighestPriorityTask()

Removes the highest priority task from the linked list by updating the head pointer to the next task. Assumes the list is not empty.

4. removeTaskById()

Removes a task by its ID from the linked list. Traverses the list to find the matching task, updates pointers to maintain list integrity, and deletes the task.

5. main()

Serves as the program's entry point, presenting a menu-driven interface for users to interact with the task management system. Handles user input and calls corresponding functions.

Challenges Faced:

During the development of the task management system, I encountered a challenge that user can enters the duplicate id which a posing a problem in removeTaskById() method as i have identical ids:

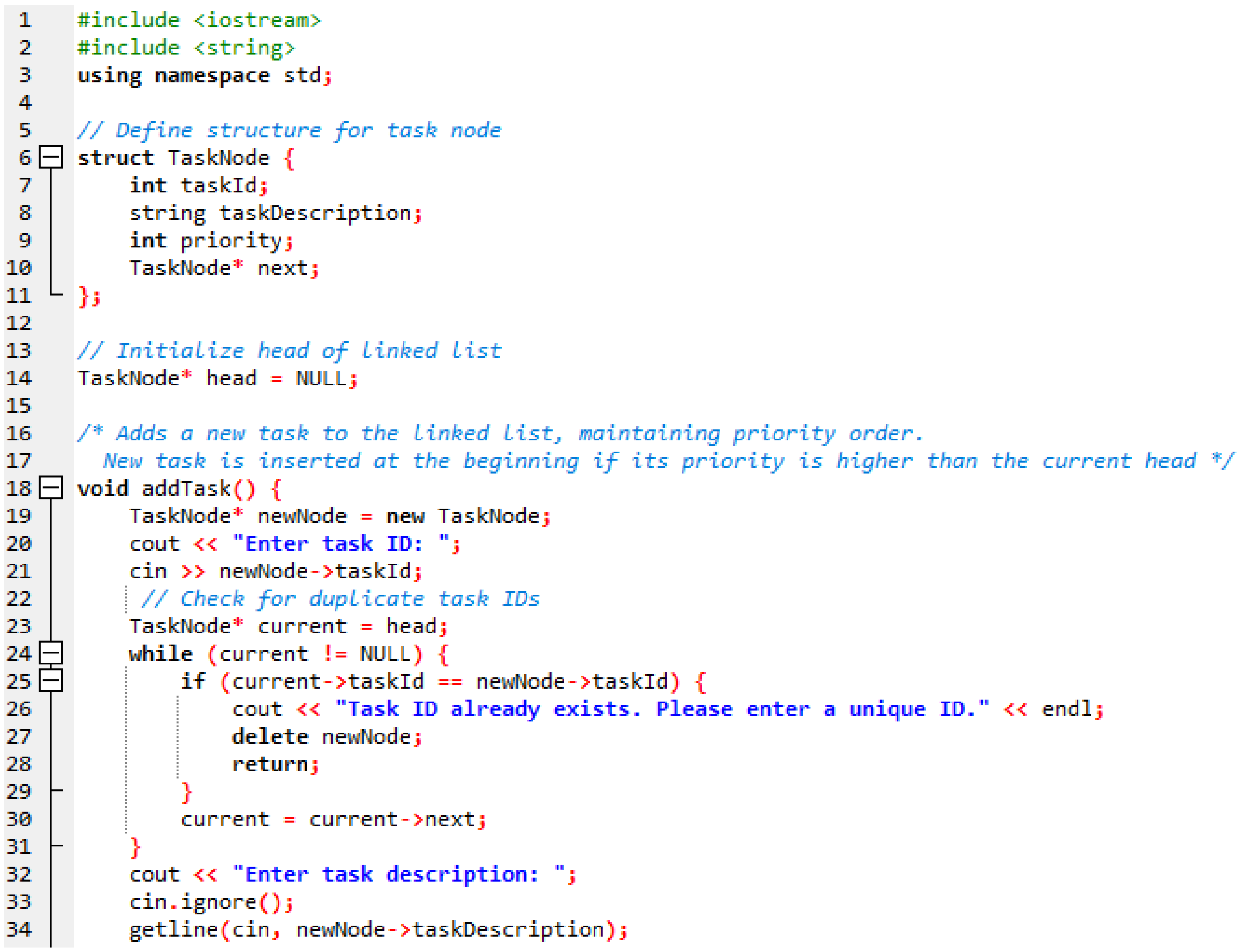
Solution:

I have Added a duplicate ID check in the addTask method to prevent users from entering identical IDs.

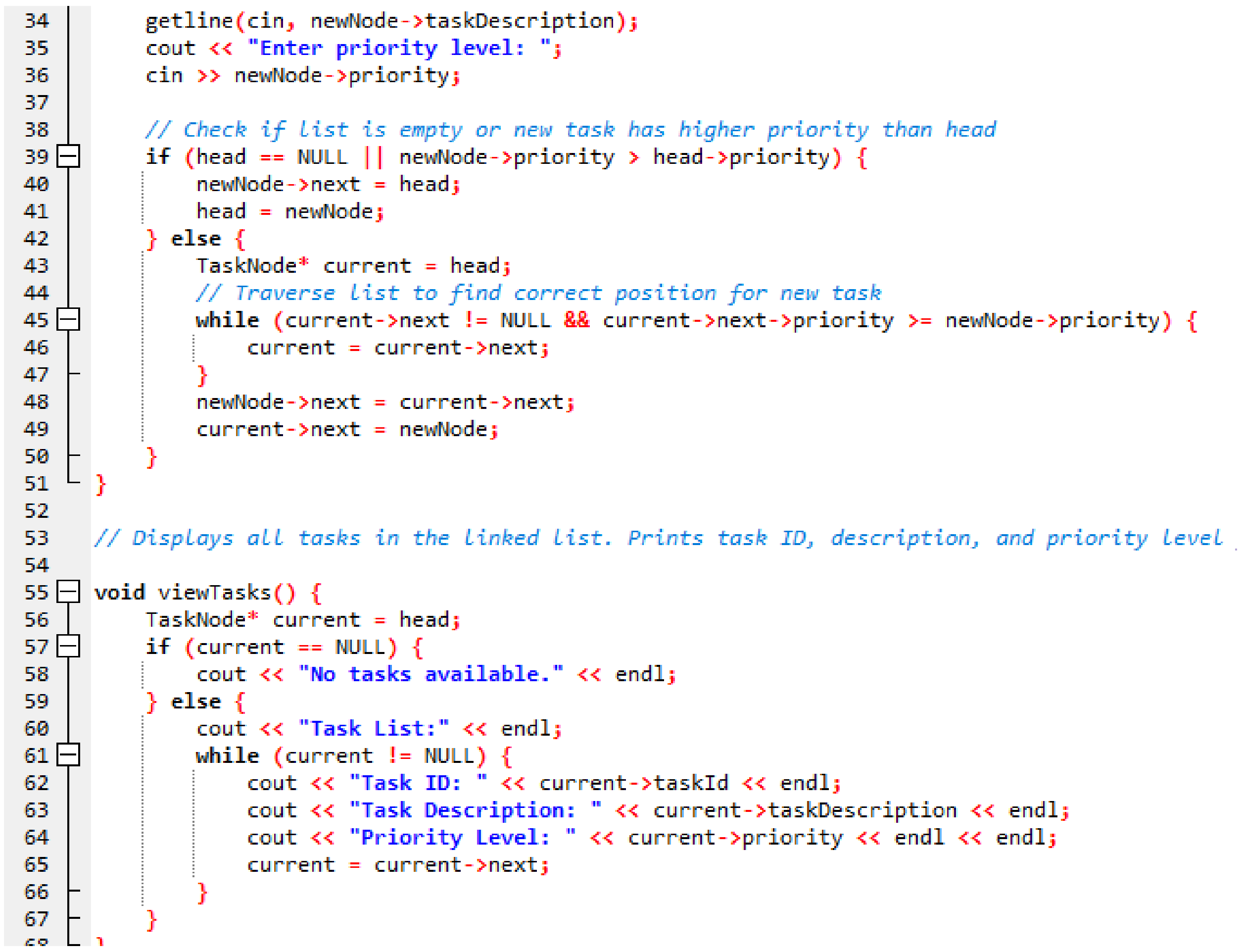
Conclusion:

Through this assignment, I gained hands-on experience implementing a singly linked list to manage tasks. Developed essential functions for task management: adding, viewing, removing, and searching tasks. This project allowed me to apply theoretical concepts to practical problems.

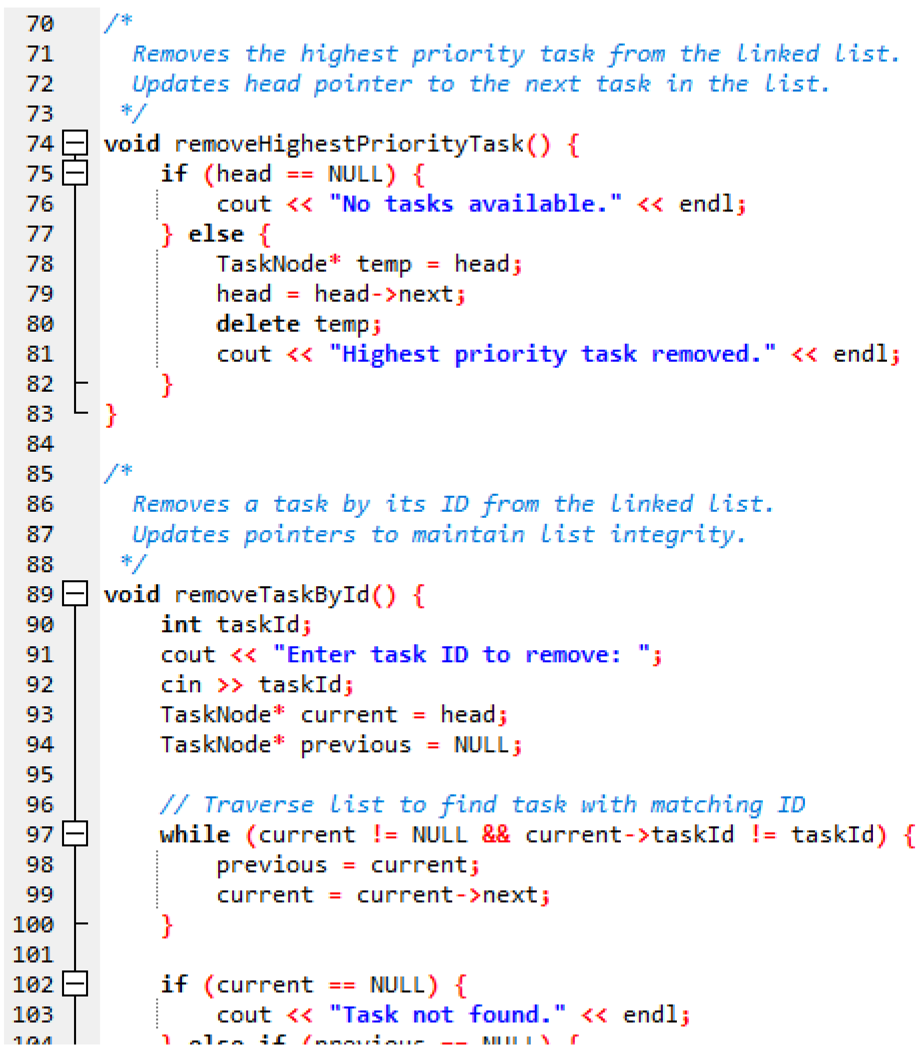
Code:



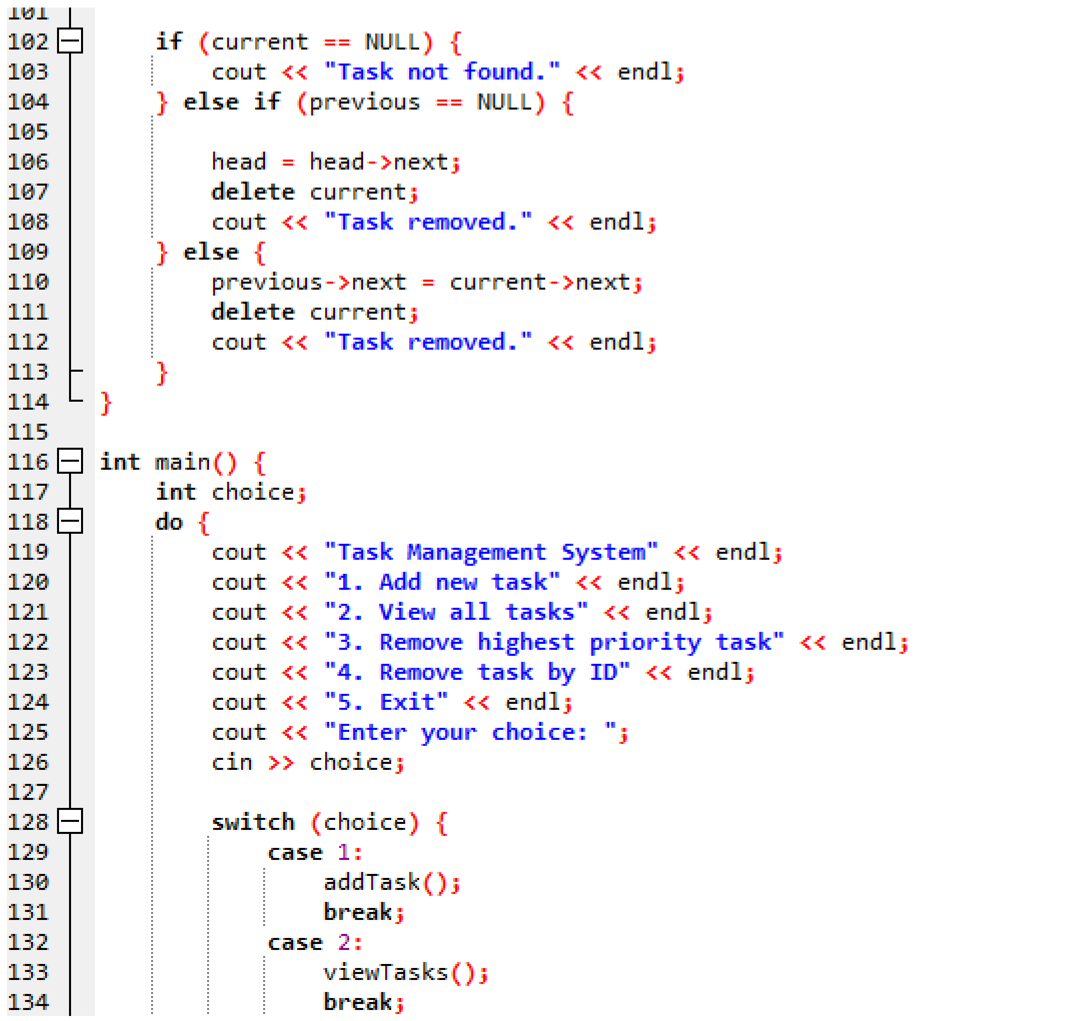
```cpp
#include <iostream>
#include <string>
using namespace std;

// Define structure for task node
struct TaskNode {
    int taskId;
    string taskDescription;
    int priority;
    TaskNode* next;
};

// Initialize head of linked list
TaskNode* head = NULL;

/* Adds a new task to the linked list, maintaining priority order.
   New task is inserted at the beginning if its priority is higher than the current head */
void addTask() {
    TaskNode* newNode = new TaskNode;
    cout << "Enter task ID: ";
    cin >> newNode->taskId;
     // Check for duplicate task IDs
    TaskNode* current = head;
    while (current != NULL) {
        if (current->taskId == newNode->taskId) {
            cout << "Task ID already exists. Please enter a unique ID." << endl;
            delete newNode;
            return;
        }
        current = current->next;
    }
    cout << "Enter task description: ";
    cin.ignore();
    getline(cin, newNode->taskDescription);
```



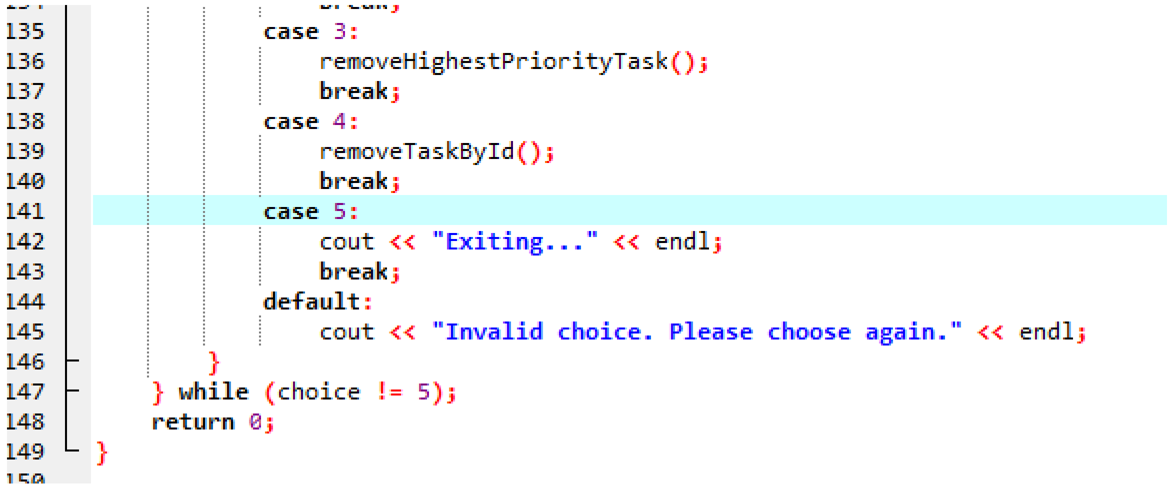
```cpp
    getline(cin, newNode->taskDescription);
    cout << "Enter priority level: ";
    cin >> newNode->priority;

    // Check if list is empty or new task has higher priority than head
    if (head == NULL || newNode->priority > head->priority) {
        newNode->next = head;
        head = newNode;
    } else {
        TaskNode* current = head;
        // Traverse list to find correct position for new task
        while (current->next != NULL && current->next->priority >= newNode->priority) {
            current = current->next;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}

// Displays all tasks in the linked list. Prints task ID, description, and priority level

void viewTasks() {
    TaskNode* current = head;
    if (current == NULL) {
        cout << "No tasks available." << endl;
    } else {
        cout << "Task List:" << endl;
        while (current != NULL) {
            cout << "Task ID: " << current->taskId << endl;
            cout << "Task Description: " << current->taskDescription << endl;
            cout << "Priority Level: " << current->priority << endl << endl;
            current = current->next;
        }
    }
}
```



```cpp
/*
  Removes the highest priority task from the linked list.
  Updates head pointer to the next task in the list.
 */
void removeHighestPriorityTask() {
    if (head == NULL) {
        cout << "No tasks available." << endl;
    } else {
        TaskNode* temp = head;
        head = head->next;
        delete temp;
        cout << "Highest priority task removed." << endl;
    }
}

/*
  Removes a task by its ID from the linked list.
  Updates pointers to maintain list integrity.
 */
void removeTaskById() {
    int taskId;
    cout << "Enter task ID to remove: ";
    cin >> taskId;
    TaskNode* current = head;
    TaskNode* previous = NULL;

    // Traverse list to find task with matching ID
    while (current != NULL && current->taskId != taskId) {
        previous = current;
        current = current->next;
    }

    if (current == NULL) {
        cout << "Task not found." << endl;
    } else if (previous == NULL) {
```

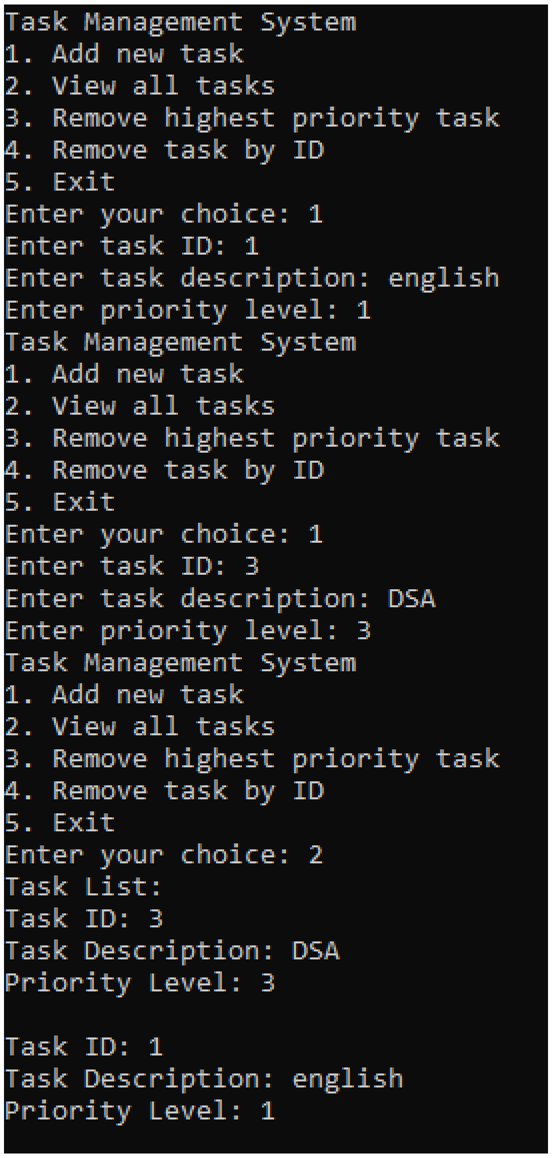


```cpp
    if (current == NULL) {
        cout << "Task not found." << endl;
    } else if (previous == NULL) {

        head = head->next;
        delete current;
        cout << "Task removed." << endl;
    } else {
        previous->next = current->next;
        delete current;
        cout << "Task removed." << endl;
    }
}

int main() {
    int choice;
    do {
        cout << "Task Management System" << endl;
        cout << "1. Add new task" << endl;
        cout << "2. View all 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:
                addTask();
                break;
            case 2:
                viewTasks();
                break;
```

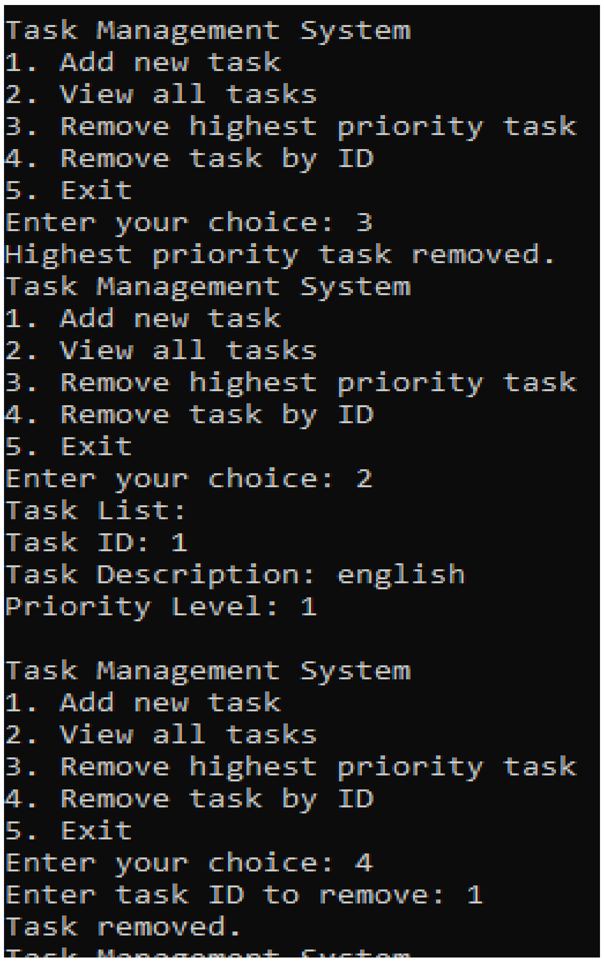


```cpp
            case 3:
                removeHighestPriorityTask();
                break;
            case 4:
                removeTaskById();
                break;
            case 5:
                cout << "Exiting..." << endl;
                break;
            default:
                cout << "Invalid choice. Please choose again." << endl;
        }
    } while (choice != 5);
    return 0;
}
```

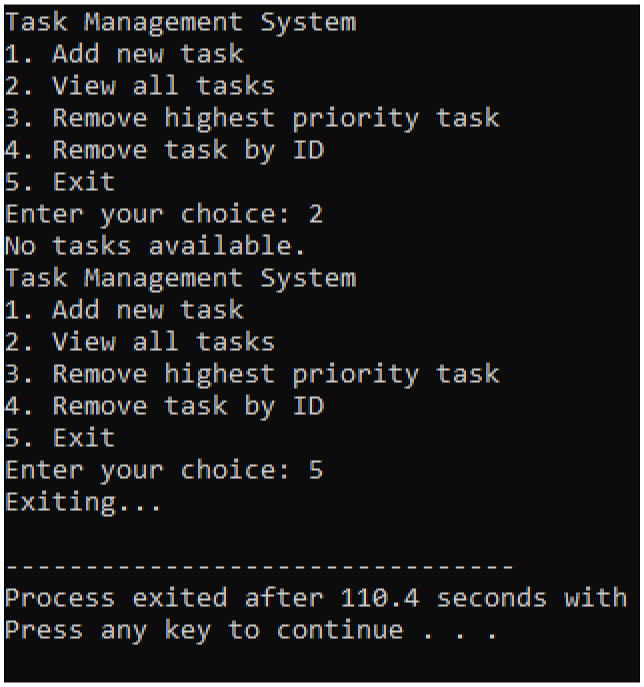
OUTPUT:



```
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 1
Enter task ID: 1
Enter task description: english
Enter priority level: 1
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 1
Enter task ID: 3
Enter task description: DSA
Enter priority level: 3
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 2
Task List:
Task ID: 3
Task Description: DSA
Priority Level: 3

Task ID: 1
Task Description: english
Priority Level: 1
```



```
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 3
Highest priority task removed.
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 2
Task List:
Task ID: 1
Task Description: english
Priority Level: 1

Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 4
Enter task ID to remove: 1
Task removed.
```



```
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 2
No tasks available.
Task Management System
1. Add new task
2. View all tasks
3. Remove highest priority task
4. Remove task by ID
5. Exit
Enter your choice: 5
Exiting...

--------------------------------
Process exited after 110.4 seconds with
Press any key to continue . . .
```