DATA STRUCTURES

THOERY ASSIGNMENT # 1

{ Task Management System }

ABEERA FAROOQ

SP23-BSE-049

24th Sep, 2024

QUESTION :

Implement the code for the assignment in a single C++ file (linked_list_operations.cpp). Ensure the code is well-documented with comments explaining the logic and functionality of each function.

SOLUTION :

Task Management System using Linked List:

Overview:

This system manages tasks using a singly linked list. Each task has an ID, description, and priority. Tasks are inserted in order of priority, with higher priority tasks at the beginning.

Key Operations:

1. **Create Task:** Creates a new task node with the given ID, description, and priority.

2. **Insert Task:** Inserts the new task into the list based on its priority. Higher priority tasks go first.
3. **Remove Highest Priority Task:** Removes the task at the beginning of the list (highest priority).
4. **Remove Task by ID:** Finds and removes a specific task based on its ID.
5. **View Tasks:** Displays all tasks in the list.

Code Structure:

- **Task Structure:** Defines the properties of a task (ID, description, priority, and a pointer to the next task).
- **Functions:**
  - createTask: Creates a new task node.
  - insertTask: Inserts a task into the list based on priority.
  - removeHighestPriorityTask: Removes the first node (highest priority).
  - removeTaskById: Searches for and removes a task by ID.
  - viewTasks: Displays all tasks in the list.
- **Main Function:** Handles user input and calls the appropriate functions.

Example Usage:

1. User adds a task with a high priority.
2. User adds another task with a lower priority.
3. System displays all tasks, showing the higher priority task first.
4. User removes the highest priority task.
5. System displays the remaining task.

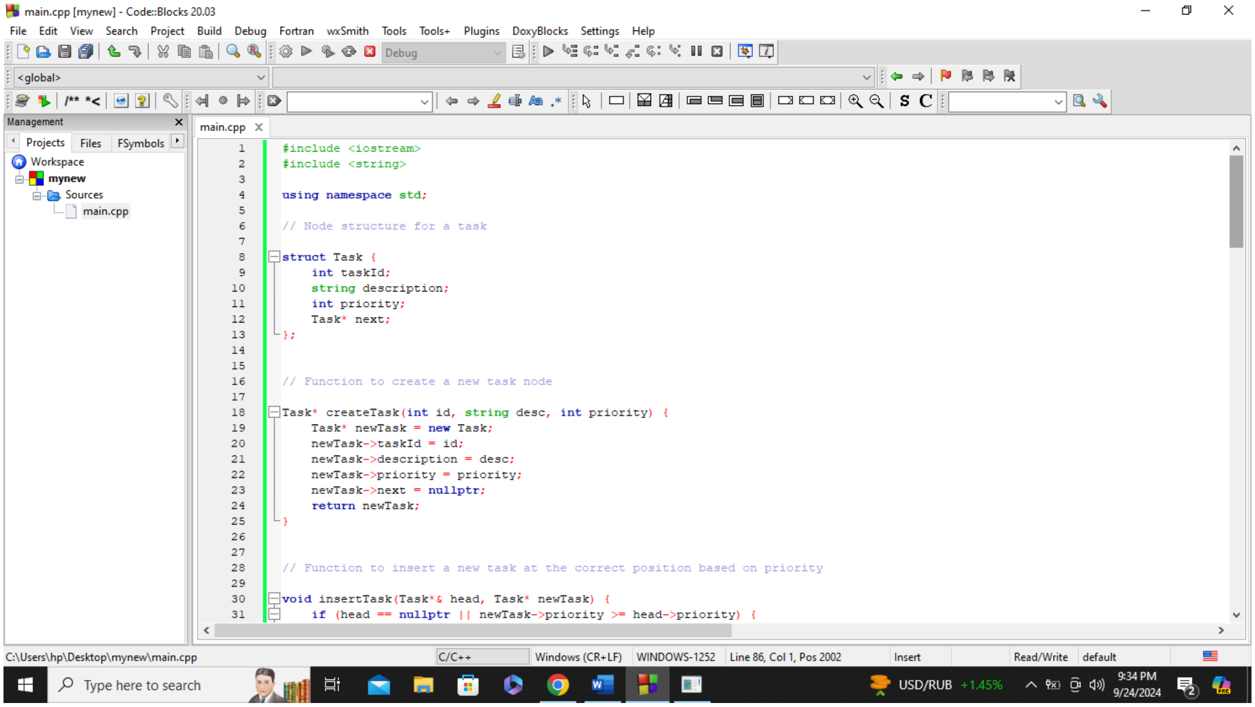
Benefits of Linked Lists:

- Efficient insertion and deletion at any position.

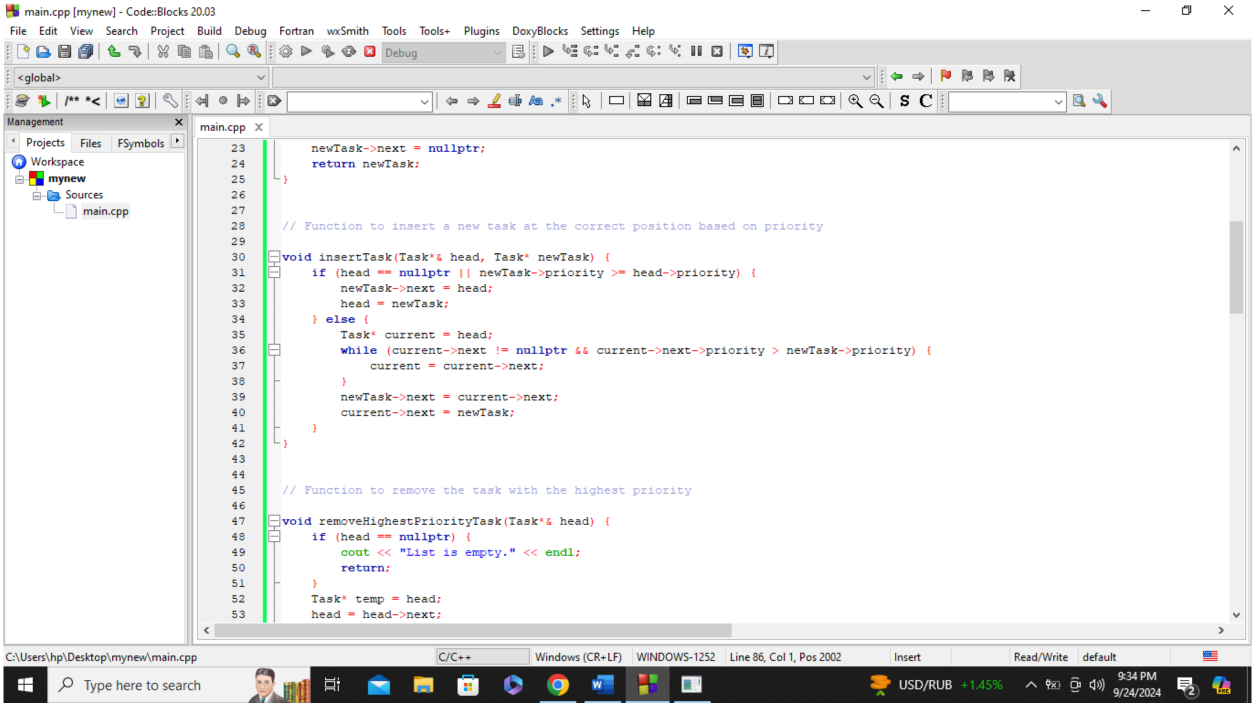
- Dynamic size.
- No need to know the size in advance.

SCREENSHOTS:

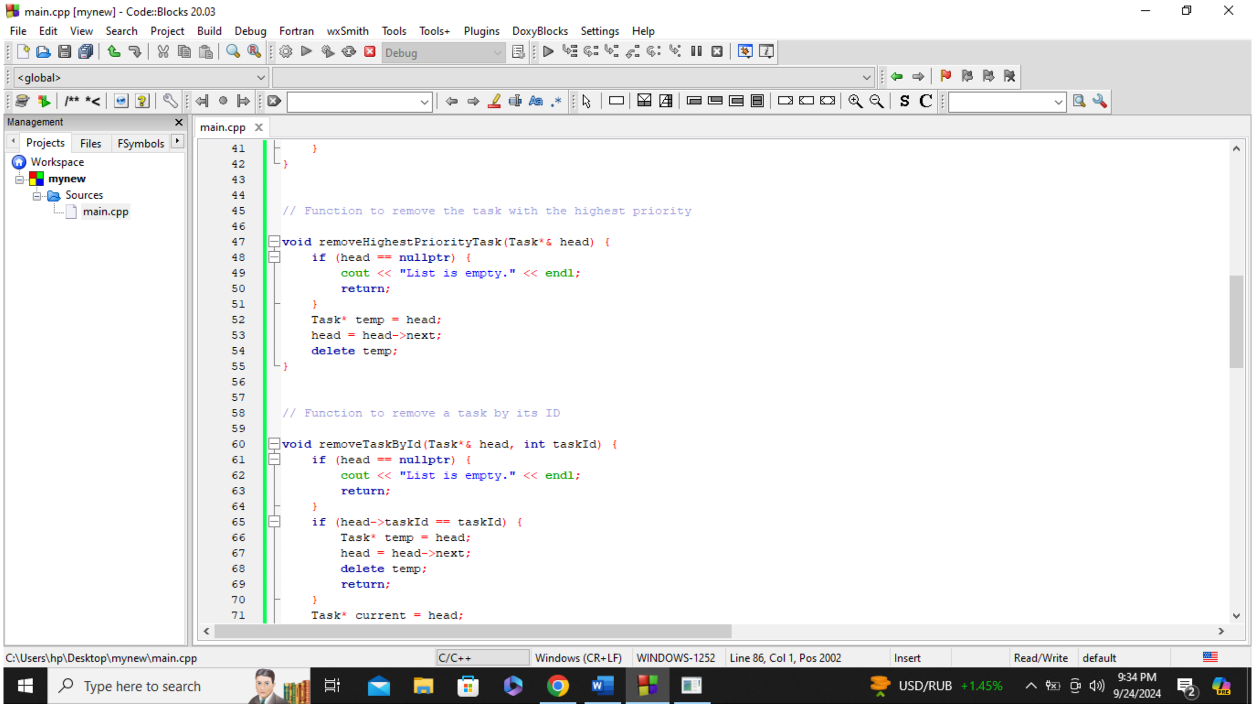
CODE;



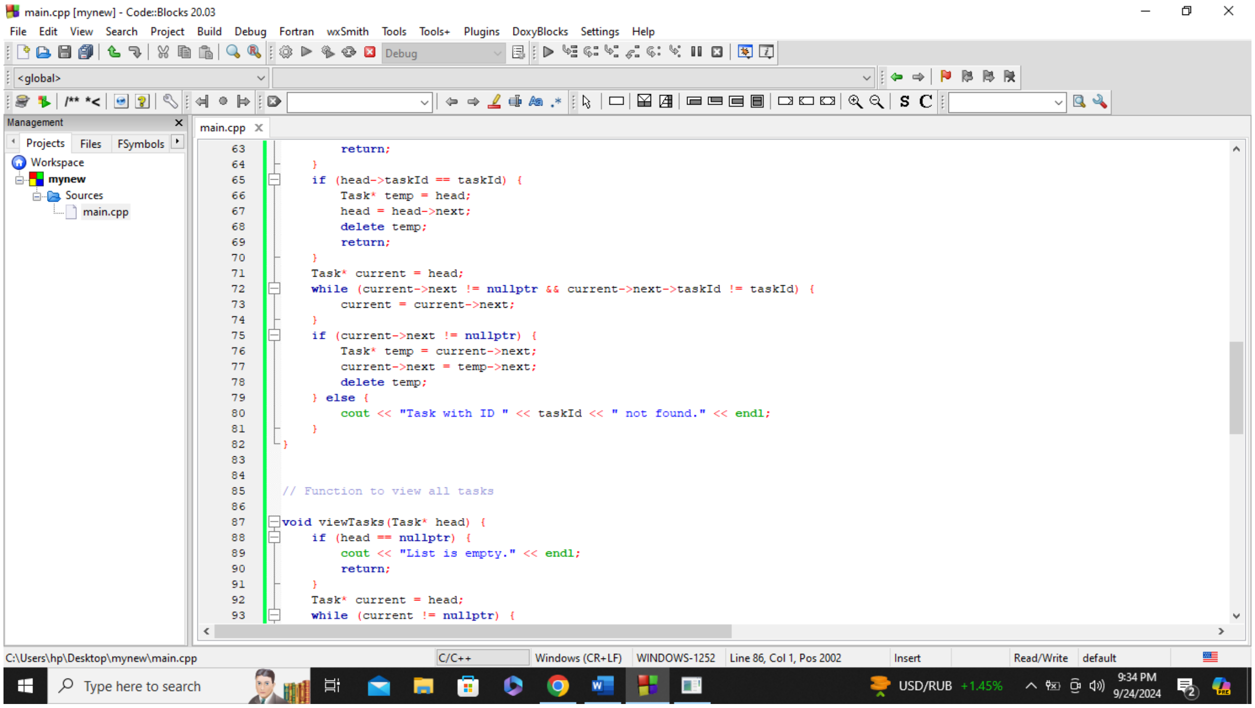
```cpp
#include <iostream>
#include <string>

using namespace std;

// Node structure for a task

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


// Function to create a new task node

Task* createTask(int id, string desc, int priority) {
    Task* newTask = new Task;
    newTask->taskId = id;
    newTask->description = desc;
    newTask->priority = priority;
    newTask->next = nullptr;
    return newTask;
}


// Function to insert a new task at the correct position based on priority

void insertTask(Task*& head, Task* newTask) {
    if (head == nullptr || newTask->priority >= head->priority) {
```



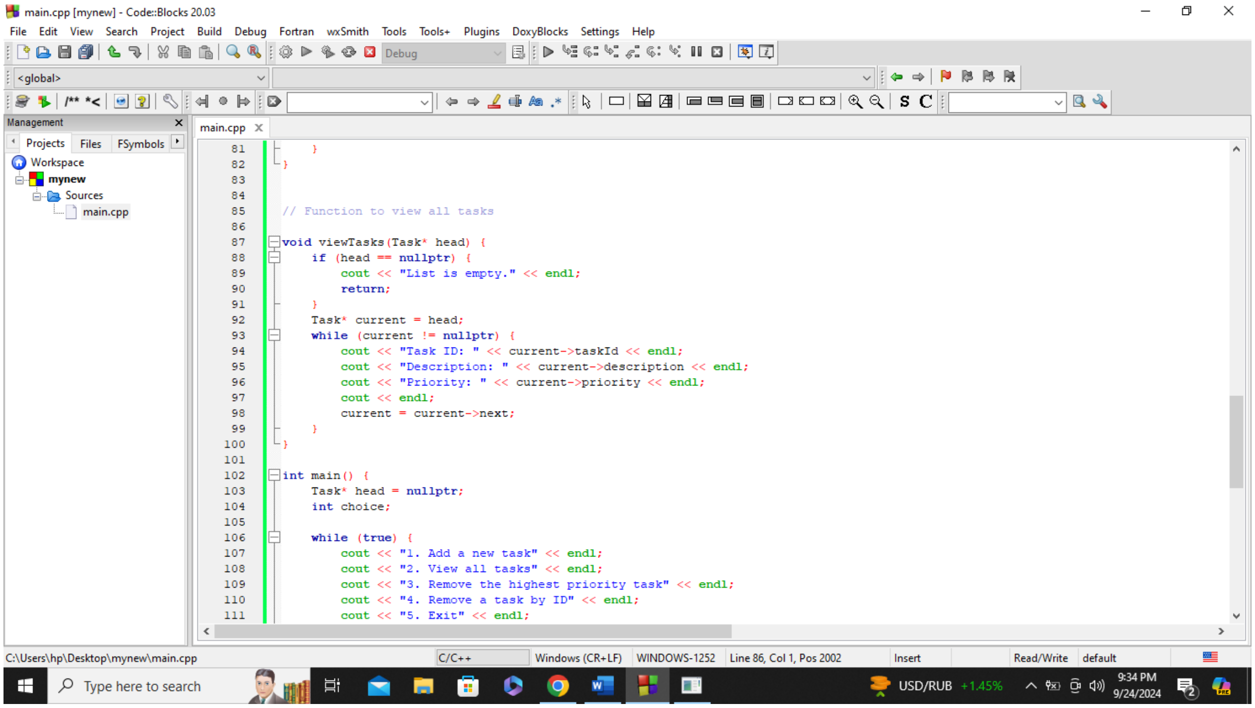
```cpp
    newTask->next = nullptr;
    return newTask;
}


// Function to insert a new task at the correct position based on priority

void insertTask(Task*& head, Task* newTask) {
    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;
    }
}


// Function to remove the task with the highest priority

void removeHighestPriorityTask(Task*& head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Task* temp = head;
    head = head->next;
```



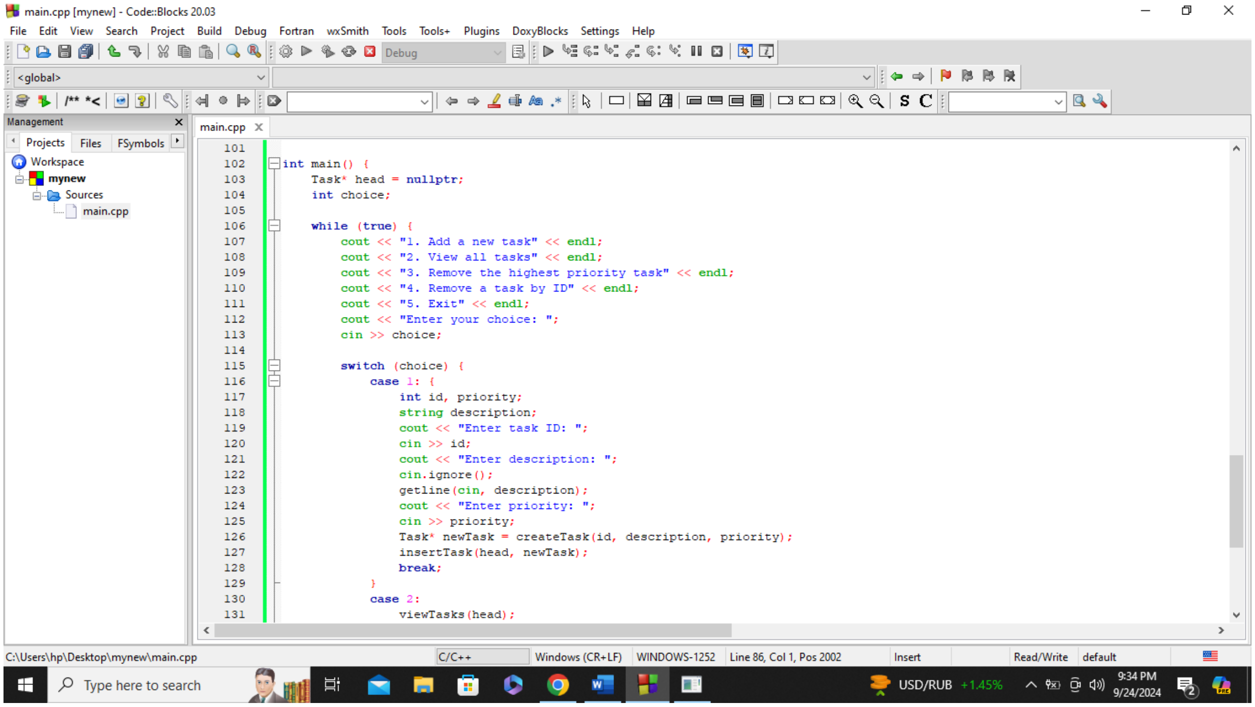
```cpp
    }
}


// Function to remove the task with the highest priority

void removeHighestPriorityTask(Task*& head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Task* temp = head;
    head = head->next;
    delete temp;
}


// Function to remove a task by its ID

void removeTaskById(Task*& head, int taskId) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    if (head->taskId == taskId) {
        Task* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Task* current = head;
```



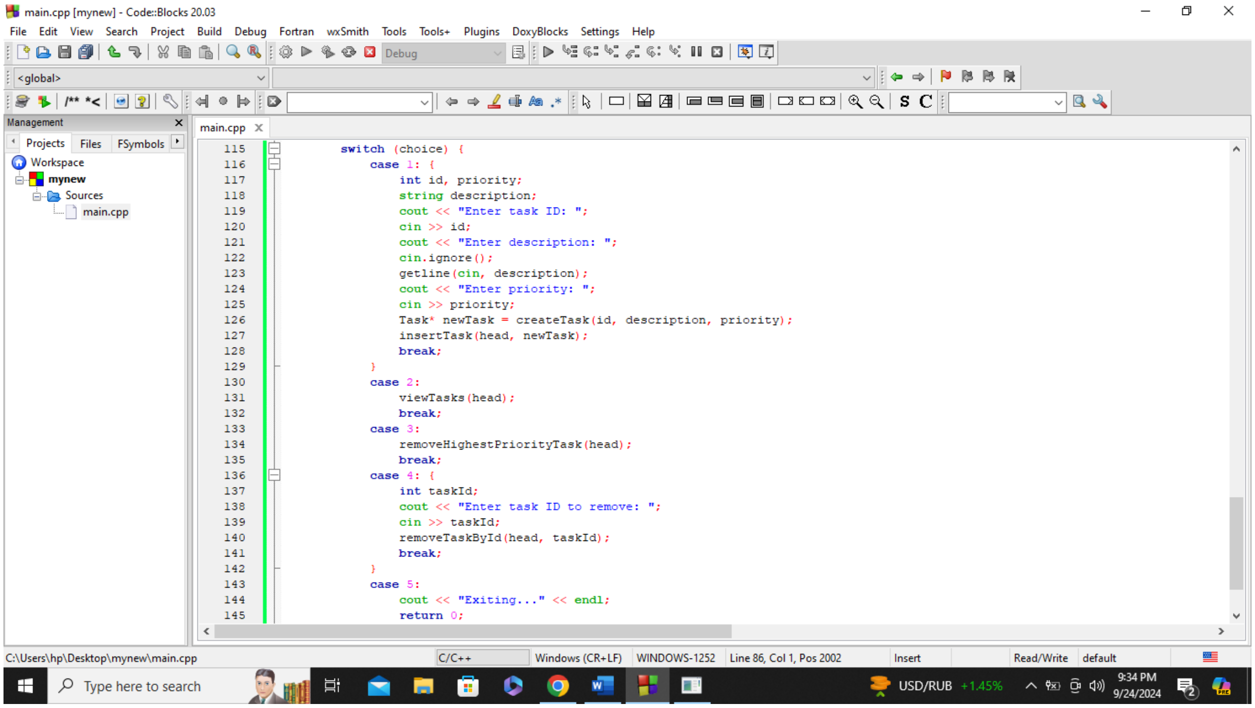
```
        return;
    }
    if (head->taskId == taskId) {
        Task* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Task* current = head;
    while (current->next != nullptr && current->next->taskId != taskId) {
        current = current->next;
    }
    if (current->next != nullptr) {
        Task* temp = current->next;
        current->next = temp->next;
        delete temp;
    } else {
        cout << "Task with ID " << taskId << " not found." << endl;
    }
}


// Function to view all tasks

void viewTasks(Task* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Task* current = head;
    while (current != nullptr) {
```



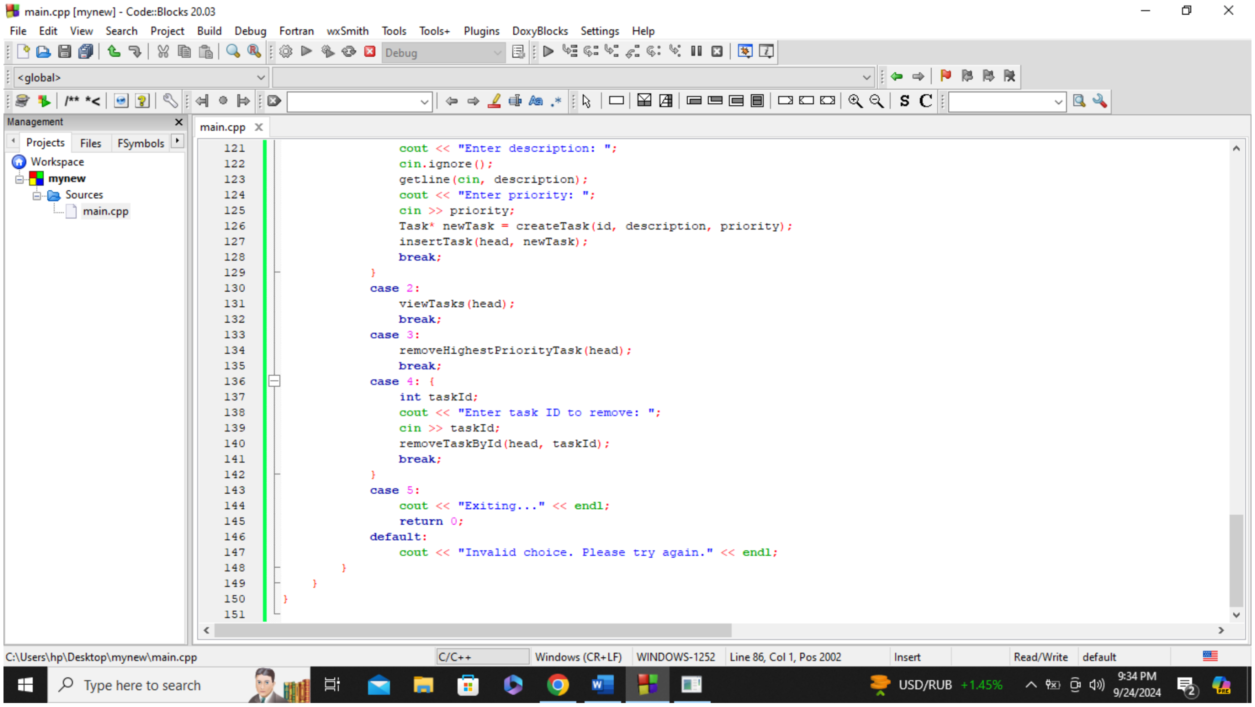
```
    }
}


// Function to view all tasks

void viewTasks(Task* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Task* current = head;
    while (current != nullptr) {
        cout << "Task ID: " << current->taskId << endl;
        cout << "Description: " << current->description << endl;
        cout << "Priority: " << current->priority << endl;
        cout << endl;
        current = current->next;
    }
}

int main() {
    Task* head = nullptr;
    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;
```


main.cpp [mynew] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files FSymbols
Workspace
mynew
Sources
main.cpp
C:\Users\hp\Desktop\mynew\main.cpp
C/C++ Windows (CR+LF) WINDOWS-1252 Line 86, Col 1, Pos 2002 Insert Read/Write default
Type here to search
USD/RUB +1.45%
9:34 PM
9/24/2024

```
int main() {
    Task* head = nullptr;
    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 description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter priority: ";
                cin >> priority;
                Task* newTask = createTask(id, description, priority);
                insertTask(head, newTask);
                break;
            }
            case 2:
                viewTasks(head);
```

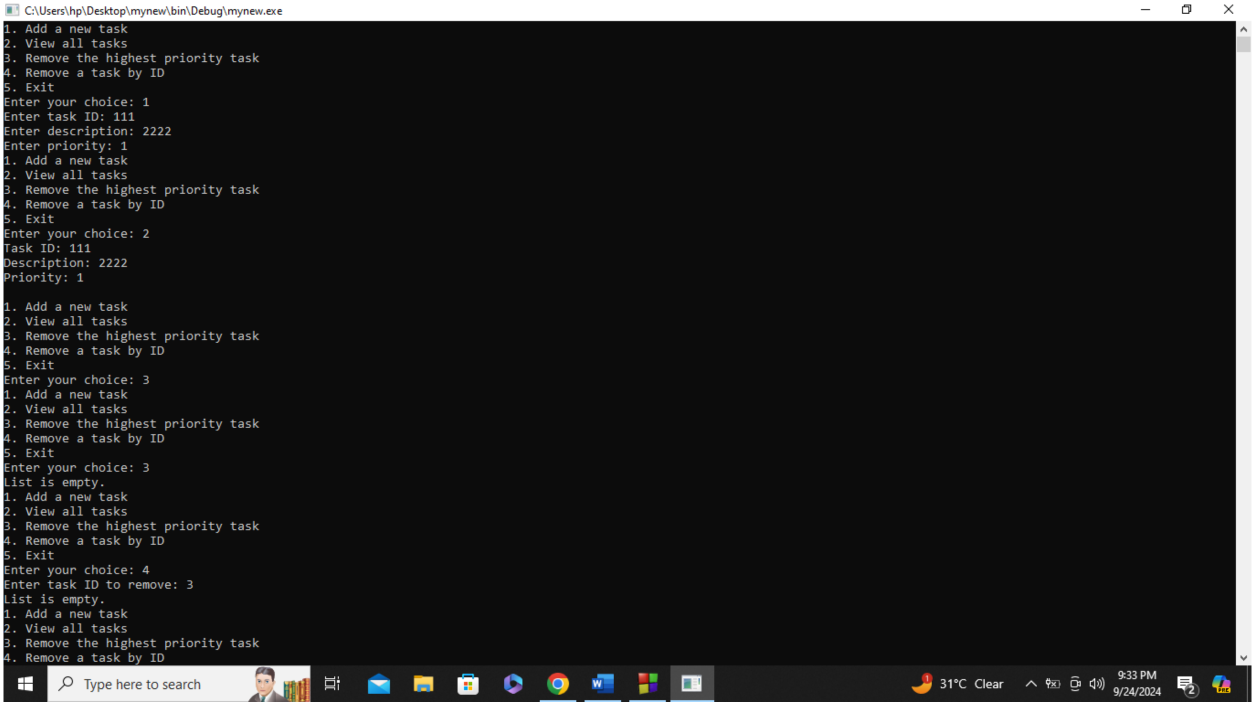

main.cpp [mynew] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files FSymbols
Workspace
mynew
Sources
main.cpp
C:\Users\hp\Desktop\mynew\main.cpp
C/C++ Windows (CR+LF) WINDOWS-1252 Line 86, Col 1, Pos 2002 Insert Read/Write default
Type here to search
USD/RUB +1.45%
9:34 PM
9/24/2024

```
        switch (choice) {
            case 1: {
                int id, priority;
                string description;
                cout << "Enter task ID: ";
                cin >> id;
                cout << "Enter description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter priority: ";
                cin >> priority;
                Task* newTask = createTask(id, description, priority);
                insertTask(head, newTask);
                break;
            }
            case 2:
                viewTasks(head);
                break;
            case 3:
                removeHighestPriorityTask(head);
                break;
            case 4: {
                int taskId;
                cout << "Enter task ID to remove: ";
                cin >> taskId;
                removeTaskById(head, taskId);
                break;
            }
            case 5:
                cout << "Exiting..." << endl;
                return 0;
```

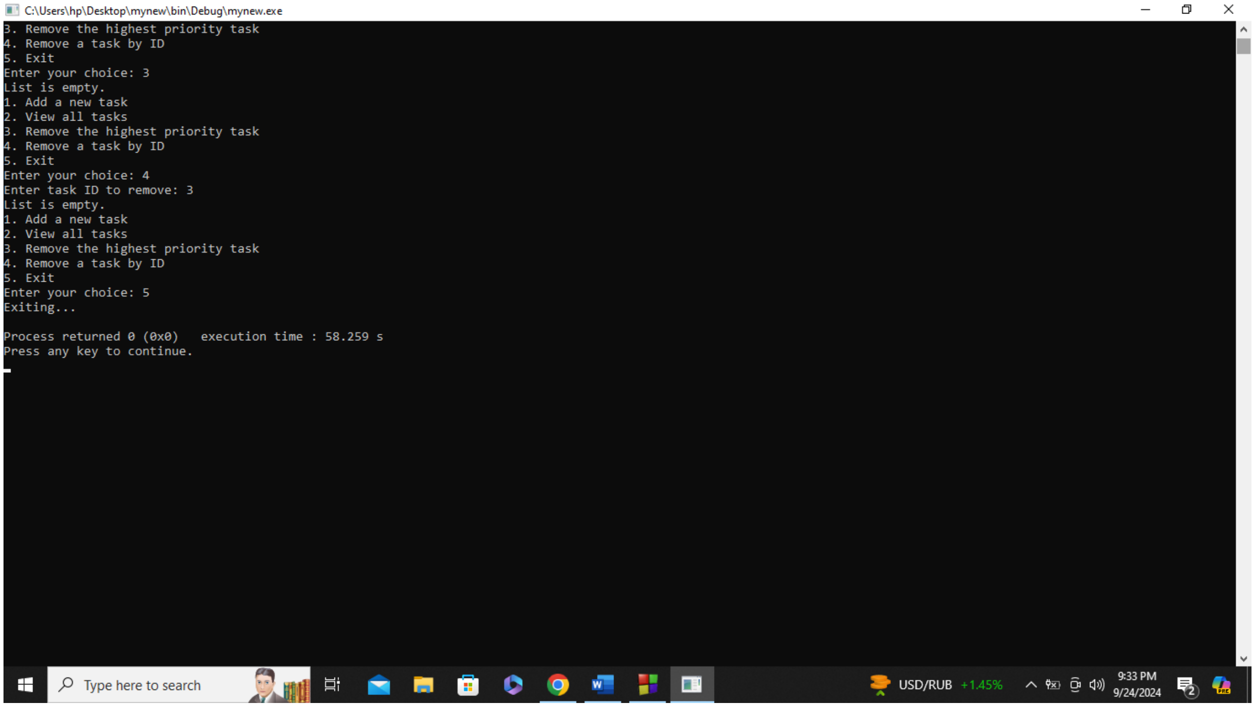


```
                cout << "Enter description: ";
                cin.ignore();
                getline(cin, description);
                cout << "Enter priority: ";
                cin >> priority;
                Task* newTask = createTask(id, description, priority);
                insertTask(head, newTask);
                break;
            }
            case 2:
                viewTasks(head);
                break;
            case 3:
                removeHighestPriorityTask(head);
                break;
            case 4: {
                int taskId;
                cout << "Enter task ID to remove: ";
                cin >> taskId;
                removeTaskById(head, taskId);
                break;
            }
            case 5:
                cout << "Exiting..." << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    }
}

```

OUTPUT;



```
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: 111
Enter description: 2222
Enter priority: 1
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: 111
Description: 2222
Priority: 1

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
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
List is empty.
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: 3
List is empty.
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
```



```
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 3
List is empty.
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: 3
List is empty.
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...

Process returned 0 (0x0)   execution time : 58.259 s
Press any key to continue.
```

Conclusion:

This task management system effectively utilizes a singly linked list to store and manipulate tasks. The system allows users to add, view, and remove tasks based on their priority, providing a flexible and efficient solution for organizing and managing tasks.