COMSATS University Islamabad, Attock Campus

Program: BSE

Course: ________________________**DSA Theory**

Azlan Aamir

Name: ________________________

Reg no: _________________________**sp23- Bse-023**

Assignment no: _________________________**01**

Date: _________________________**22 September 2024**

Submitted to: _______________________**Mr. Muhammad kamran**

Objective: The program implements a task management system using a singly linked list, allowing users to manage tasks based on their priorities and IDs

Operations Implemented:

1. **Add a New Task**: Users can add tasks to the list, automatically placing them in the correct position based on priority.
2. **View All Tasks**: Users can view the list of current tasks along with their details.
3. **Remove Highest Priority Task**: Users can remove the task with the highest priority, which is always located at the head of the list.
4. **Remove a Task by ID**: Users can remove a specific task by its unique ID.
5. **Exit**: Users can exit the program safely.

Logic:

- D**ata Structure**: A singly linked list is used to manage tasks, where each task is represented as a node containing a unique ID, description, and priority.

- **Adding Tasks**: When a new task is added, it is inserted in the correct position based on priority. Higher-priority tasks are placed at the front of the list.

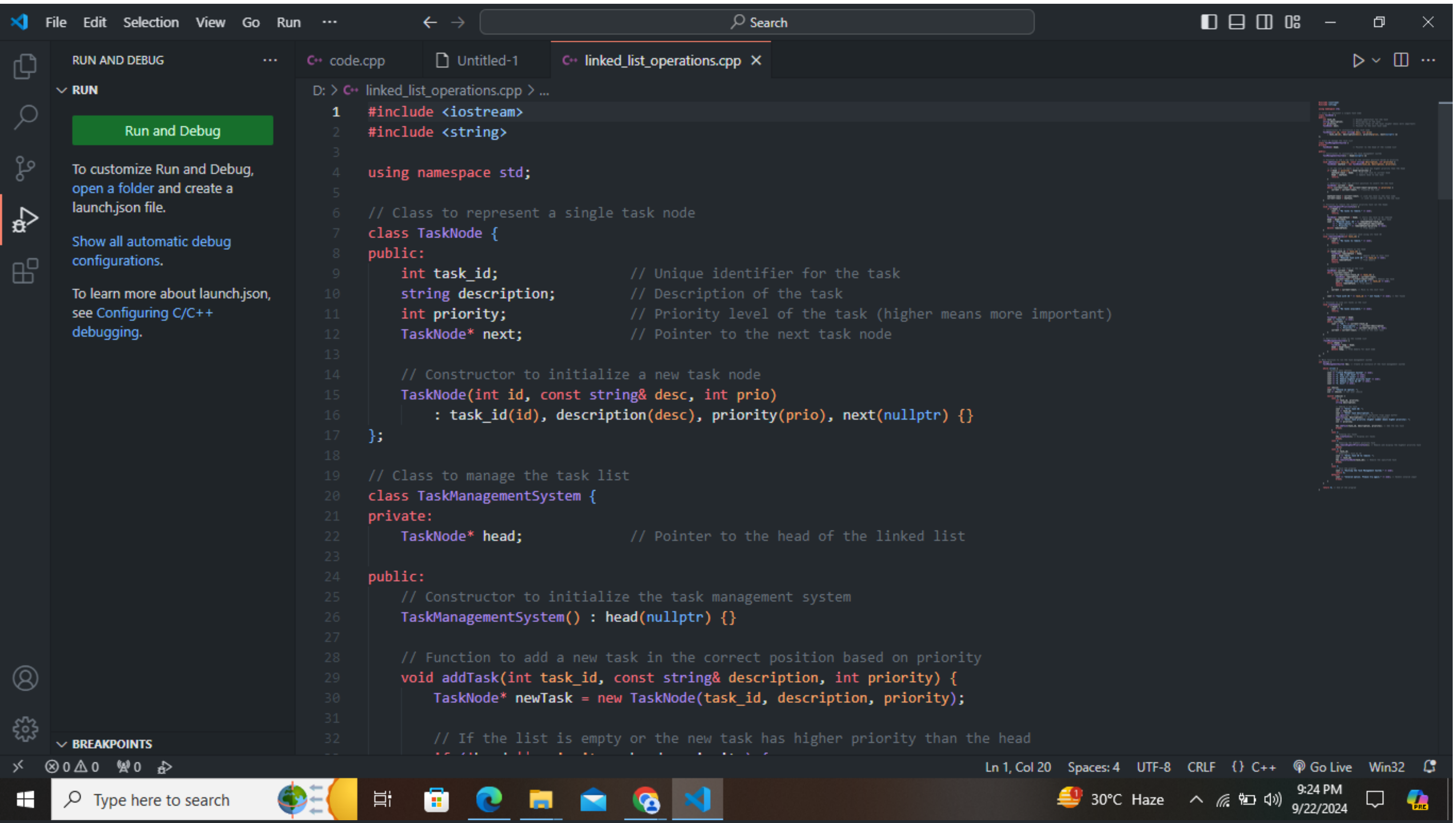
- **Removing Tasks**:

  - The highest priority task (at the head of the list) can be removed easily.
  - A specific task can be removed using its unique ID, which involves searching through the list.

- **Viewing Tasks**: Users can view all tasks in the list, displaying each task's details in order of priority.

- **Memory Management**: The system properly frees memory for removed tasks and cleans up all nodes when the program exits to prevent memory leaks.

Code:

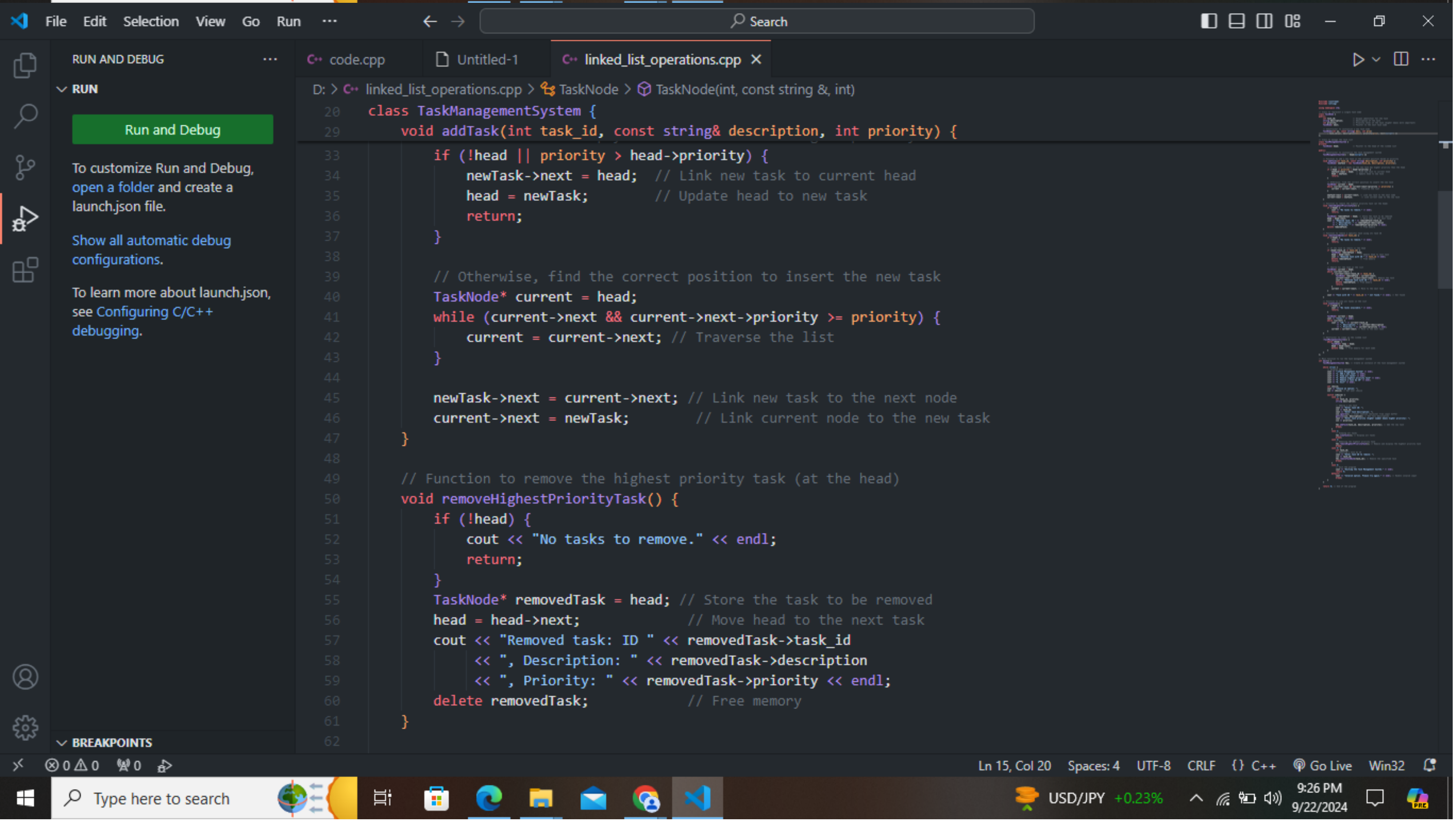


To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

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

using namespace std;

// Class to represent a single task node
class TaskNode {
public:
    int task_id;                // Unique identifier for the task
    string description;         // Description of the task
    int priority;               // Priority level of the task (higher means more important)
    TaskNode* next;             // Pointer to the next task node

    // Constructor to initialize a new task node
    TaskNode(int id, const string& desc, int prio)
        : task_id(id), description(desc), priority(prio), next(nullptr) {}
};

// Class to manage the task list
class TaskManagementSystem {
private:
    TaskNode* head;             // Pointer to the head of the linked list

public:
    // Constructor to initialize the task management system
    TaskManagementSystem() : head(nullptr) {}

    // Function to add a new task in the correct position based on priority
    void addTask(int task_id, const string& description, int priority) {
        TaskNode* newTask = new TaskNode(task_id, description, priority);

        // If the list is empty or the new task has higher priority than the head
```

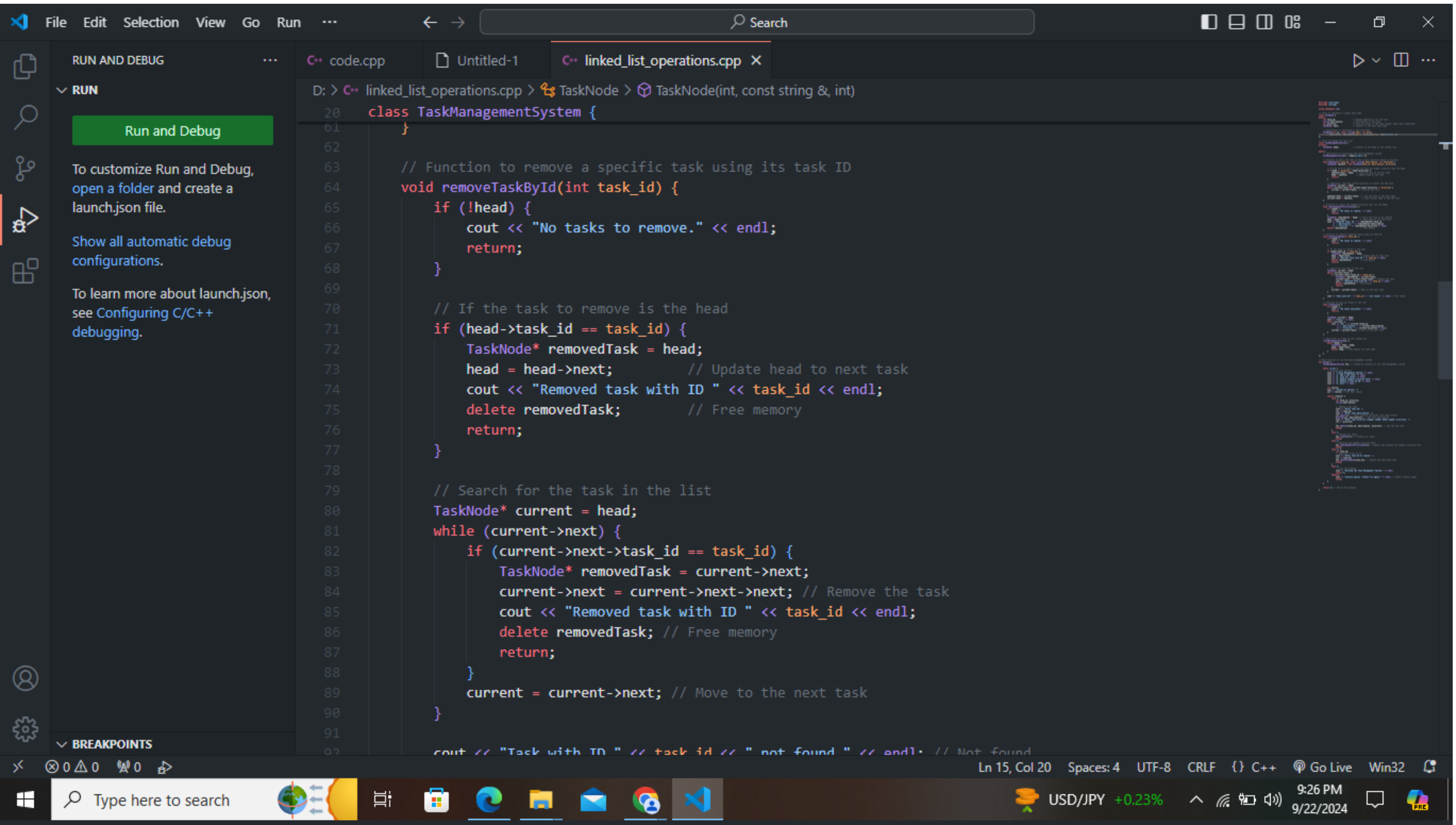


To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
class TaskManagementSystem {
    void addTask(int task_id, const string& description, int priority) {
        if (!head || priority > head->priority) {
            newTask->next = head;  // Link new task to current head
            head = newTask;        // Update head to new task
            return;
        }

        // Otherwise, find the correct position to insert the new task
        TaskNode* current = head;
        while (current->next && current->next->priority >= priority) {
            current = current->next; // Traverse the list
        }

        newTask->next = current->next; // Link new task to the next node
        current->next = newTask;        // Link current node to the new task
    }

    // Function to remove the highest priority task (at the head)
    void removeHighestPriorityTask() {
        if (!head) {
            cout << "No tasks to remove." << endl;
            return;
        }
        TaskNode* removedTask = head; // Store the task to be removed
        head = head->next;             // Move head to the next task
        cout << "Removed task: ID " << removedTask->task_id
             << ", Description: " << removedTask->description
             << ", Priority: " << removedTask->priority << endl;
        delete removedTask;            // Free memory
    }

```

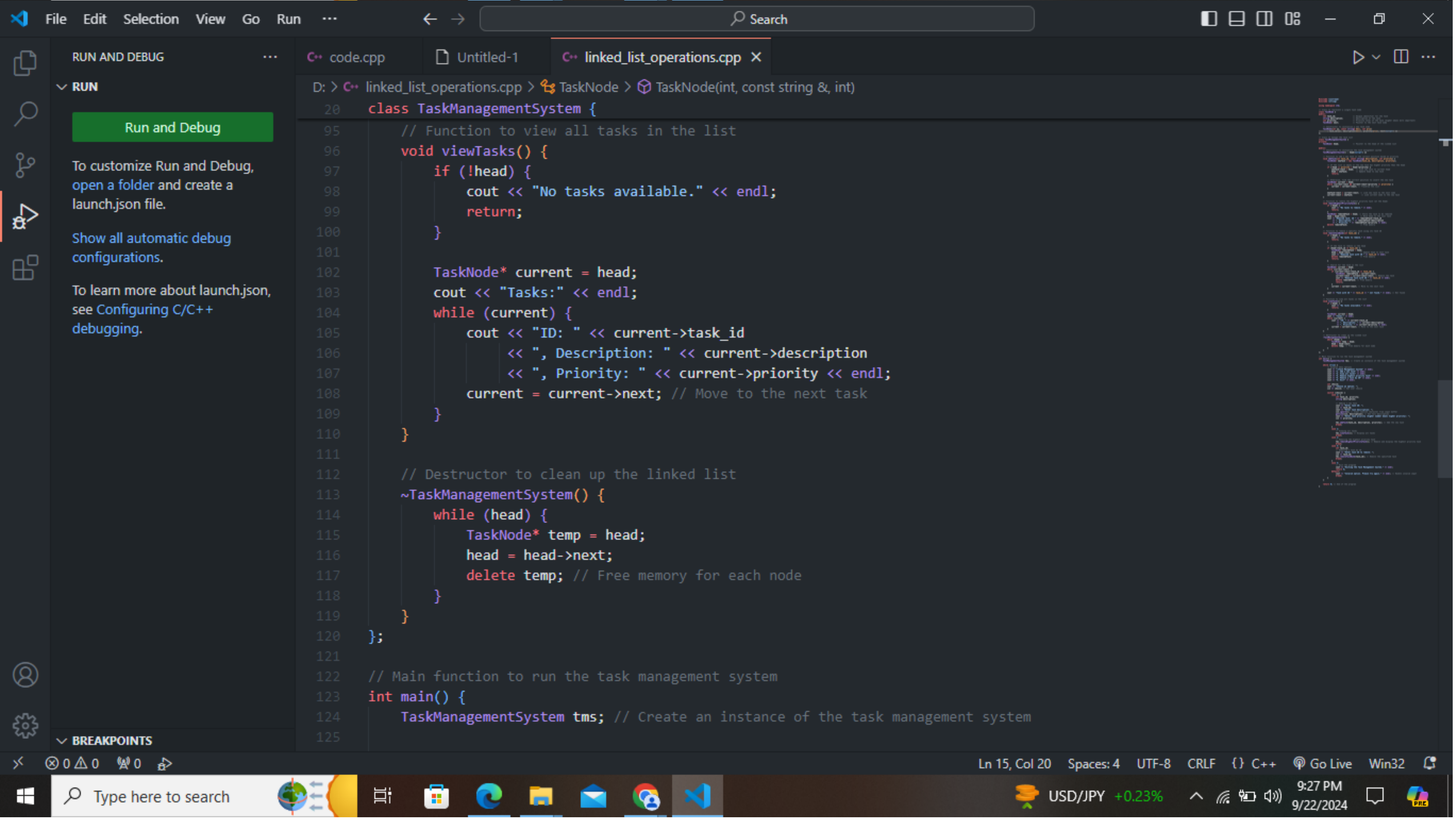


To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
class TaskManagementSystem {
    }

    // Function to remove a specific task using its task ID
    void removeTaskById(int task_id) {
        if (!head) {
            cout << "No tasks to remove." << endl;
            return;
        }

        // If the task to remove is the head
        if (head->task_id == task_id) {
            TaskNode* removedTask = head;
            head = head->next;          // Update head to next task
            cout << "Removed task with ID " << task_id << endl;
            delete removedTask;         // Free memory
            return;
        }

        // Search for the task in the list
        TaskNode* current = head;
        while (current->next) {
            if (current->next->task_id == task_id) {
                TaskNode* removedTask = current->next;
                current->next = current->next->next; // Remove the task
                cout << "Removed task with ID " << task_id << endl;
                delete removedTask; // Free memory
                return;
            }
            current = current->next; // Move to the next task
        }

        cout << "Task with ID " << task_id << " not found." << endl; // Not found
```

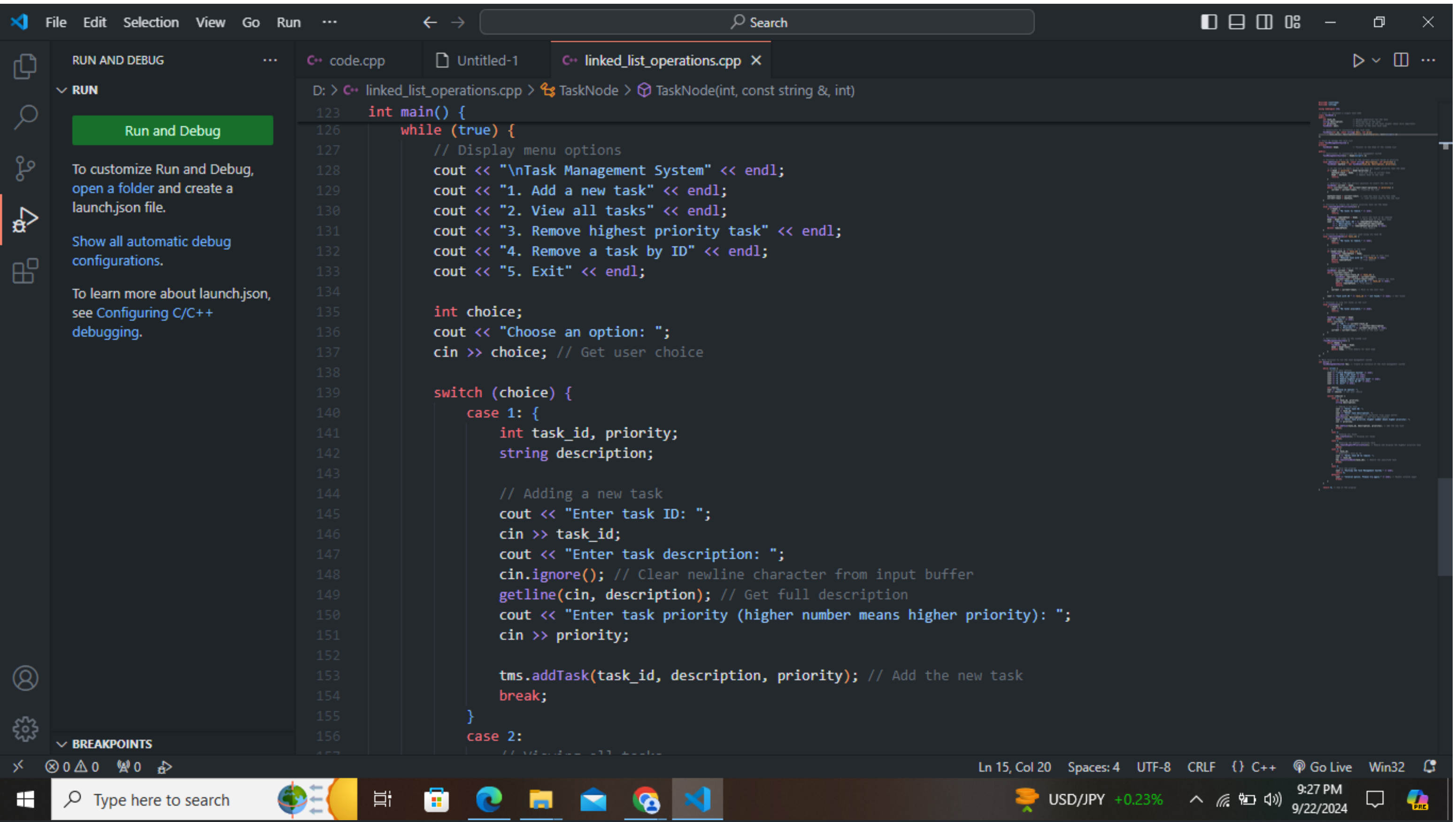


To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
class TaskManagementSystem {
    // Function to view all tasks in the list
    void viewTasks() {
        if (!head) {
            cout << "No tasks available." << endl;
            return;
        }

        TaskNode* current = head;
        cout << "Tasks:" << endl;
        while (current) {
            cout << "ID: " << current->task_id
                 << ", Description: " << current->description
                 << ", Priority: " << current->priority << endl;
            current = current->next; // Move to the next task
        }
    }

    // Destructor to clean up the linked list
    ~TaskManagementSystem() {
        while (head) {
            TaskNode* temp = head;
            head = head->next;
            delete temp; // Free memory for each node
        }
    }
};

// Main function to run the task management system
int main() {
    TaskManagementSystem tms; // Create an instance of the task management system

```

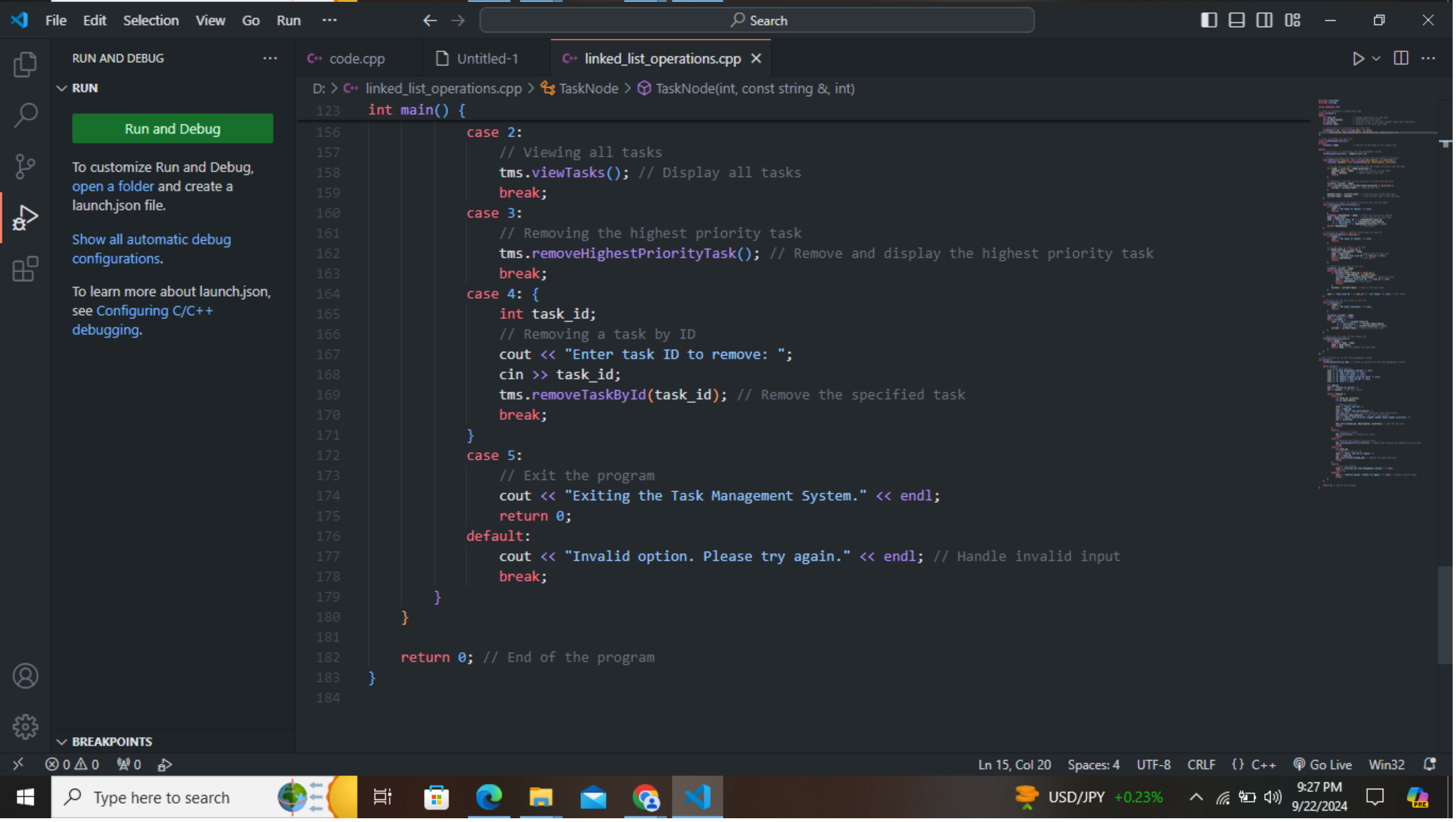

File Edit Selection View Go Run ···
Search
RUN AND DEBUG
RUN
Run and Debug
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
BREAKPOINTS
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:27 PM
9/22/2024

To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
int main() {
    while (true) {
        // Display menu options
        cout << "\nTask Management System" << endl;
        cout << "1. Add a new task" << endl;
        cout << "2. View all tasks" << endl;
        cout << "3. Remove highest priority task" << endl;
        cout << "4. Remove a task by ID" << endl;
        cout << "5. Exit" << endl;

        int choice;
        cout << "Choose an option: ";
        cin >> choice; // Get user choice

        switch (choice) {
            case 1: {
                int task_id, priority;
                string description;

                // Adding a new task
                cout << "Enter task ID: ";
                cin >> task_id;
                cout << "Enter task description: ";
                cin.ignore(); // Clear newline character from input buffer
                getline(cin, description); // Get full description
                cout << "Enter task priority (higher number means higher priority): ";
                cin >> priority;

                tms.addTask(task_id, description, priority); // Add the new task
                break;
            }
            case 2:
```

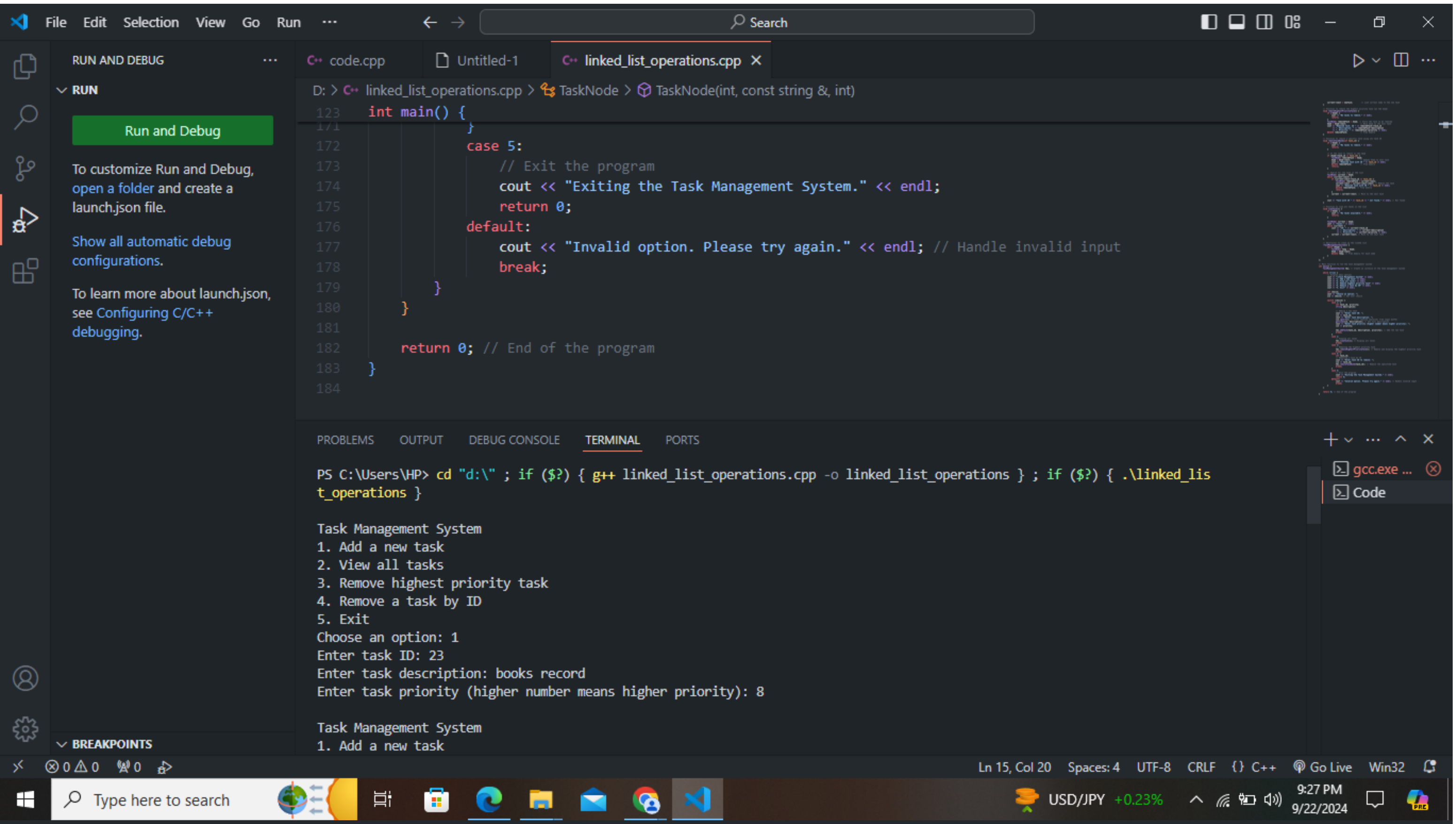

File Edit Selection View Go Run ···
Search
RUN AND DEBUG
RUN
Run and Debug
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
BREAKPOINTS
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:27 PM
9/22/2024

To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

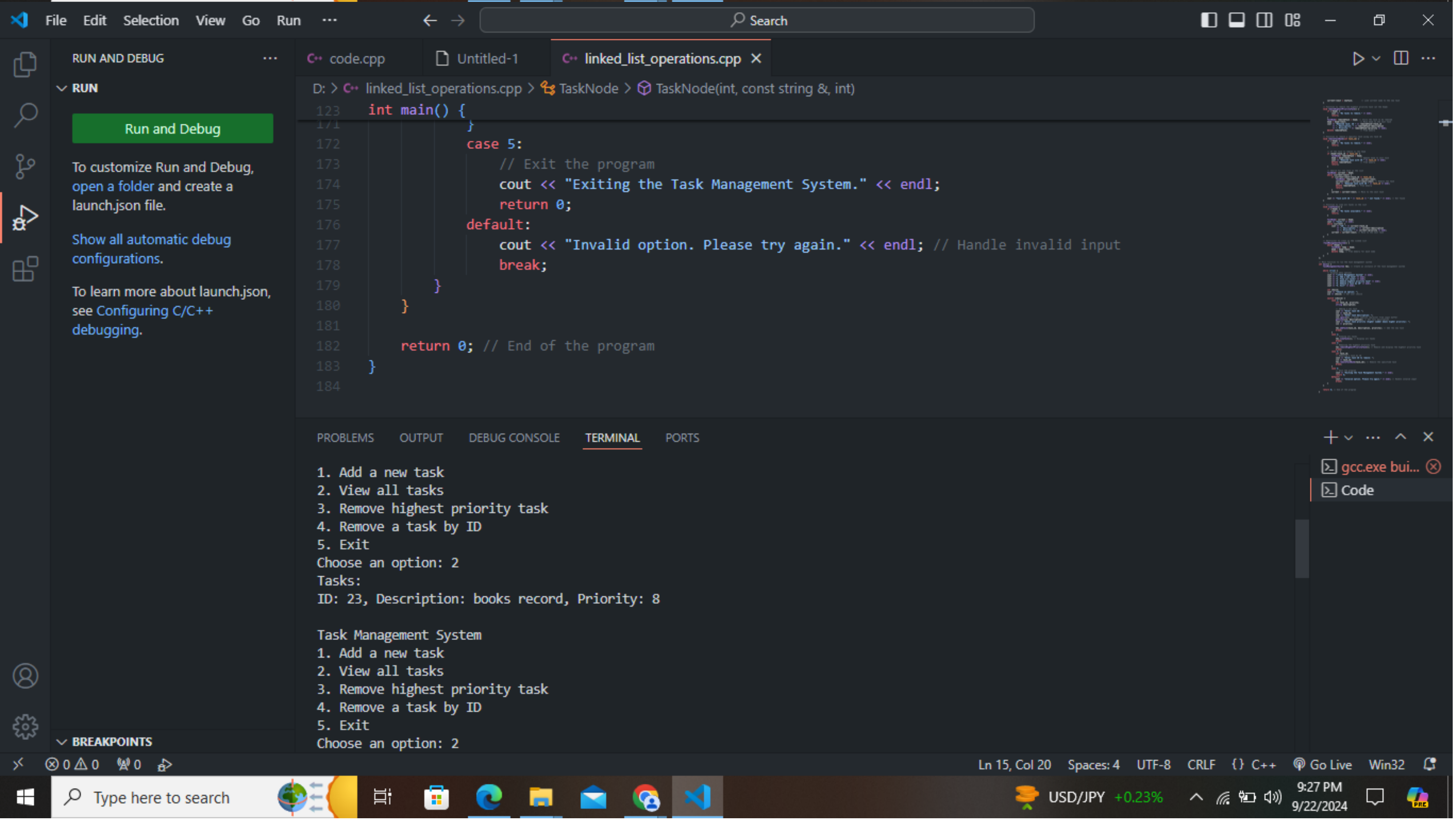
To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
int main() {
            case 2:
                // Viewing all tasks
                tms.viewTasks(); // Display all tasks
                break;
            case 3:
                // Removing the highest priority task
                tms.removeHighestPriorityTask(); // Remove and display the highest priority task
                break;
            case 4: {
                int task_id;
                // Removing a task by ID
                cout << "Enter task ID to remove: ";
                cin >> task_id;
                tms.removeTaskById(task_id); // Remove the specified task
                break;
            }
            case 5:
                // Exit the program
                cout << "Exiting the Task Management System." << endl;
                return 0;
            default:
                cout << "Invalid option. Please try again." << endl; // Handle invalid input
                break;
        }
    }

    return 0; // End of the program
}
```


File Edit Selection View Go Run ···
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
BREAKPOINTS
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gcc.exe ...
Code
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:27 PM
9/22/2024

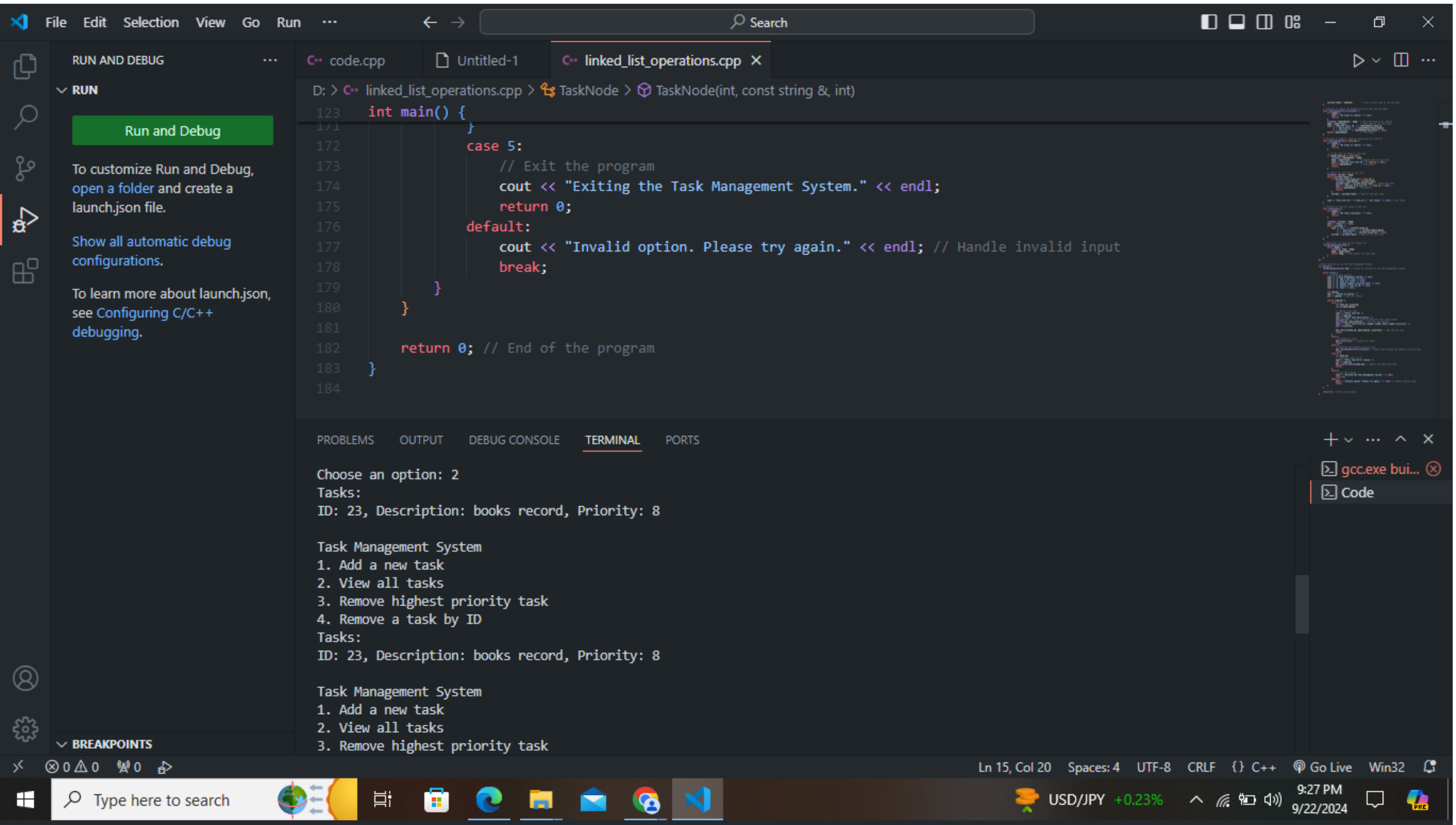
```
int main() {
            }
            case 5:
                // Exit the program
                cout << "Exiting the Task Management System." << endl;
                return 0;
            default:
                cout << "Invalid option. Please try again." << endl; // Handle invalid input
                break;
        }
    }

    return 0; // End of the program
}
```

```
PS C:\Users\HP> cd "d:\" ; if ($?) { g++ linked_list_operations.cpp -o linked_list_operations } ; if ($?) { .\linked_lis
t_operations }

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
Choose an option: 1
Enter task ID: 23
Enter task description: books record
Enter task priority (higher number means higher priority): 8

Task Management System
1. Add a new task
```


File Edit Selection View Go Run ···
Search
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
BREAKPOINTS
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gcc.exe bui...
Code
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:27 PM
9/22/2024

```
int main() {
            }
            case 5:
                // Exit the program
                cout << "Exiting the Task Management System." << endl;
                return 0;
            default:
                cout << "Invalid option. Please try again." << endl; // Handle invalid input
                break;
        }
    }

    return 0; // End of the program
}
```

```
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
Choose an option: 2
Tasks:
ID: 23, Description: books record, Priority: 8

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
Choose an option: 2
```


File Edit Selection View Go Run ···
Search
RUN AND DEBUG ···
RUN
Run and Debug
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gcc.exe bui...
Code
BREAKPOINTS
⊗ 0 ⚠ 0 ₩ 0
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:27 PM
9/22/2024

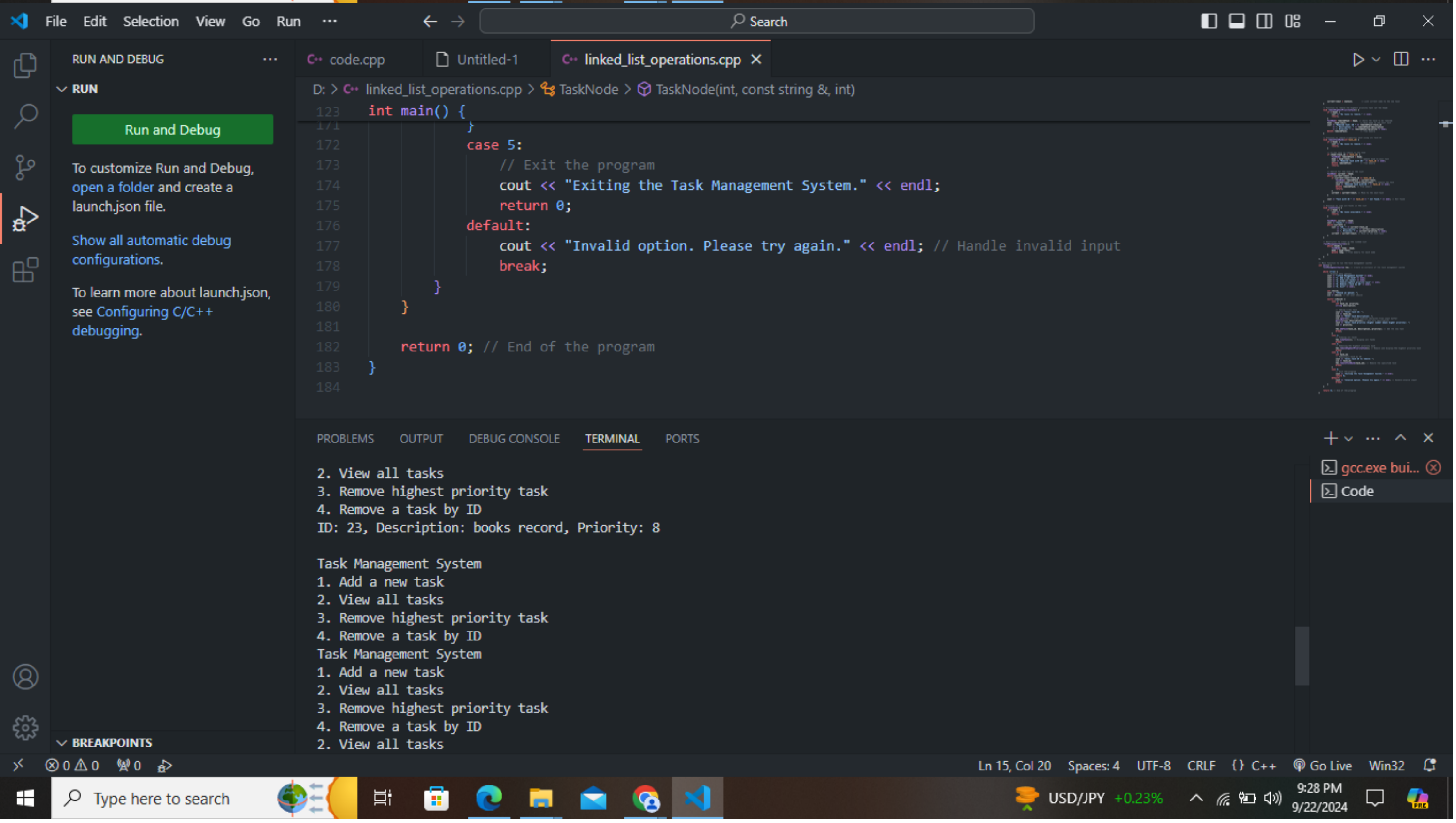
To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
int main() {
            }
            case 5:
                // Exit the program
                cout << "Exiting the Task Management System." << endl;
                return 0;
            default:
                cout << "Invalid option. Please try again." << endl; // Handle invalid input
                break;
        }
    }

    return 0; // End of the program
}
```

```
Choose an option: 2
Tasks:
ID: 23, Description: books record, Priority: 8

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
Tasks:
ID: 23, Description: books record, Priority: 8

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
```


File Edit Selection View Go Run ···
Search
RUN AND DEBUG ···
RUN
Run and Debug
code.cpp
Untitled-1
linked_list_operations.cpp
D: > linked_list_operations.cpp > TaskNode > TaskNode(int, const string &, int)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gcc.exe bui...
Code
BREAKPOINTS
⊗ 0 ⚠ 0 ₩ 0
Ln 15, Col 20 Spaces: 4 UTF-8 CRLF {} C++ Go Live Win32
Type here to search
USD/JPY +0.23%
9:28 PM
9/22/2024

To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

To learn more about launch.json, see Configuring C/C++ debugging.

```cpp
int main() {
            }
            case 5:
                // Exit the program
                cout << "Exiting the Task Management System." << endl;
                return 0;
            default:
                cout << "Invalid option. Please try again." << endl; // Handle invalid input
                break;
        }
    }

    return 0; // End of the program
}
```

```
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
ID: 23, Description: books record, Priority: 8

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
2. View all tasks
```

Conclusions: Through this assignment, I learned how to implement a basic task management system using a singly linked list in C++. It enhanced my understanding of data structures, particularly linked lists, and reinforced concepts like encapsulation and dynamic memory management.