Task Management

```
#include <iostream>

#include <string>

using namespace std;

class Task {

private:

   static int lastId;

   int id;

   string description, status, duedate;

public:

   Task(const string& desc, const string& stat, const string& due) {

      id = ++lastId;

      description = desc;

      status = stat;

      duedate = due;

   }

   int getId() const { return id; }

   string getDescription() const { return description; }

   string getStatus() const { return status; }

   string getDueDate() const { return duedate; }

};

int Task::lastId = 0;

class Node {

public:

   Task task;
```

```
    Node* previous;
    Node* next;
    Node(const Task& t) : task(t), previous(nullptr), next(nullptr) {}
};
class TaskManager {
private:
    Node* head;
    Node* next;
public:
    TaskManager() : head(nullptr), next(nullptr) {}
    void addTask(const string& desc, const string& status, const string& due) {
        Node* p = new Node(Task(desc, status, due));
        if (head == nullptr) { head = next = p; }
        else {
            next->next = p;
            p->previous = next;
            next = p;
        }
    }
    void displayTasks() {
        if (head == nullptr) { cout << "No tasks to display." << endl; }
        else {
            Node* p = head;
            cout << "Tasks:\n" << endl;
            while (p != nullptr) {
```

```
            cout << "Task ID: " << p->task.getId() << endl;
            cout << "Description: " << p->task.getDescription() << endl;
            cout << "Status: " << p->task.getStatus() << endl;
            cout << "Due Date: " << p->task.getDueDate() << endl;
            cout << endl;
            p = p->next;
        }
    }
}
void removeTask(int taskId) {
    Node* p = head;
    while (p != nullptr) {
        if (p->task.getId() == taskId) {
            if (p == head) {
                head = p->next;
                if (head != nullptr) {
                    head->previous = nullptr;
                } else {
                    next = nullptr;
                }
            } else if (p == next) {
                next = p->previous;
                next->next = nullptr;
            } else {
                p->previous->next = p->next;
```

```
            p->next->previous = p->previous;
        }
        delete p;
        cout << "Task with ID " << taskId << " removed successfully." << endl;
        return;
      }
      p = p->next;
    }
    cout << "Task with ID " << taskId << " not found." << endl;
  }
};
int main() {
  TaskManager taskManager;
  int choice, taskID;
  string desc, status, dueDate;
  do {
    cout << "\n1. Add Task" << endl;
    cout << "2. Remove Task" << endl;
    cout << "3. Display Task" << endl;
    cout << "4. Exit" << endl;
    cout << "\nEnter choice: ";
    cin >> choice;
    cout << endl;
    switch (choice) {
    case 1:
```

```
            cout << "Enter Description: ";
            cin >> desc;
            cout << "Enter Status: ";
            cin >> status;
            cout << "Enter due date: ";
            cin >> dueDate;
            taskManager.addTask(desc, status, dueDate);
            break;
        case 2:
            cout << "Enter Task ID: ";
            cin >> taskID;
            taskManager.removeTask(taskID);
            break;
        case 3: taskManager.displayTasks(); break;
        case 4: exit(0); break;
        default: cout << "\nENTER VALID CHOICE!!" << endl; break;
        }
    } while (choice != 4);
    return 0;
}
```

```
1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 1

Enter Description: cleanning
Enter Status: pending
Enter due date: 12-2-2022

1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 1

Enter Description: removeTheFlowers
Enter Status: pending
Enter due date: 2-2-2012

1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 3

Tasks:

Task ID: 1
Description: cleanning
Status: pending
Due Date: 12-2-2022

Task ID: 2
Description: removeTheFlowers
Status: pending
Due Date: 2-2-2012


1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 2

Enter Task ID: 2
Task with ID 2 removed successfully.
```

```
1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 3

Tasks:

Task ID: 1
Description: cleanning
Status: pending
Due Date: 12-2-2022


1. Add Task
2. Remove Task
3. Display Task
4. Exit

Enter choice: 4
```