

TASK 4

TO-DO LIST

❖ CODE:-

```
// C++ Program to implement the to do list

#include <iostream>

#include <string>

#include <vector>


using namespace std;


// Define Task class

class Task {

private:

    string name; // Task name

    string description; // Task description

    string dueDate; // Task due date

    bool completed; // Task completion status


public:

    // Constructor to initialize a task

    Task(const string& name, const string& description,

        const string& dueDate)

        : name(name)

        , description(description)

        , dueDate(dueDate)

        , completed(false)

    {

    }

}
```

```
// Getter for task name
```

```
string getName() const { return name; }
```

```
// Getter for task description
```

```
string getDescription() const { return description; }
```

```
// Getter for task due date
```

```
string getDueDate() const { return dueDate; }
```

```
// Getter for task completion status
```

```
bool isCompleted() const { return completed; }
```

```
// Setter for task name
```

```
void setName(const string& name) { this->name = name; }
```

```
// Setter for task description
```

```
void setDescription(const string& description)
```

```
{
```

```
    this->description = description;
```

```
}
```

```
// Setter for task due date
```

```
void setDueDate(const string& dueDate)
```

```
{
```

```
    this->dueDate = dueDate;
```

```
}
```

```
// Mark the task as completed
```

```

void markCompleted() { completed = true; }

// Display task details
void displayTask() const
{
    cout << name << " ("
        << (completed ? "Completed" : "Pending")
        << ") - Due: " << dueDate << endl
        << "  Description: " << description << endl;
}
};

```

```

// Define ToDoList class
class ToDoList {
private:
    vector<Task> tasks; // List of tasks

public:
    // Display the menu
    void displayMenu()
    {
        cout
            << "\n----- To-Do List Menu ----- \n";

        cout << "1. Add Task\n";
        cout << "2. Delete Task\n";
        cout << "3. Display Tasks\n";
        cout << "4. Mark Task as Completed\n";
        cout << "5. Edit Task\n";
        cout << "6. Exit\n";
    }
};

```

```

        cout << "-----"
            "\n";
    }

// Add a new task
void addTask()
{
    string name, description, dueDate;
    cout << "Enter task name: ";
    cin.ignore();
    getline(cin, name);
    cout << "Enter task description: ";
    getline(cin, description);
    cout << "Enter task due date (YYYY-MM-DD): ";
    getline(cin, dueDate);

    tasks.emplace_back(name, description, dueDate);
    cout << "Task added successfully!" << endl;
}

// Delete a task
void deleteTask()
{
    if (tasks.empty()) {
        cout << "No tasks to delete!" << endl;
        return;
    }

    cout << "Tasks:" << endl;
    for (int i = 0; i < tasks.size(); ++i) {

```

```

        cout << i + 1 << ". " << tasks[i].getName()
            << endl;
    }
    cout << "Enter the task number to delete: ";
    int taskNumber;
    cin >> taskNumber;
    if (taskNumber >= 1 && taskNumber <= tasks.size()) {
        tasks.erase(tasks.begin() + taskNumber - 1);
        cout << "Task deleted successfully!" << endl;
    }
    else {
        cout << "Invalid task number!" << endl;
    }
}

// Display all tasks
void displayTasks()
{
    if (tasks.empty()) {
        cout << "No tasks to display!" << endl;
        return;
    }
    cout << "Tasks:" << endl;
    for (int i = 0; i < tasks.size(); ++i) {
        cout << i + 1 << ". ";
        tasks[i].displayTask();
    }
}

```

```

// Mark a task as completed
void markTaskCompleted()
{
    if (tasks.empty()) {
        cout << "No tasks to mark as completed!"
            << endl;
        return;
    }
    cout << "Tasks:" << endl;
    for (int i = 0; i < tasks.size(); ++i) {
        cout << i + 1 << ". " << tasks[i].getName()
            << endl;
    }
    cout << "Enter the task number to mark as "
        << "completed: ";
    int taskNumber;
    cin >> taskNumber;
    if (taskNumber >= 1 && taskNumber <= tasks.size()) {
        tasks[taskNumber - 1].markCompleted();
        cout << "Task marked as completed!" << endl;
    }
    else {
        cout << "Invalid task number!" << endl;
    }
}

// Edit a task
void editTask()
{

```

```

if (tasks.empty()) {
    cout << "No tasks to edit!" << endl;
    return;
}

cout << "Tasks:" << endl;
for (int i = 0; i < tasks.size(); ++i) {
    cout << i + 1 << ". " << tasks[i].getName()
        << endl;
}

cout << "Enter the task number to edit: ";
int taskNumber;
cin >> taskNumber;

if (taskNumber >= 1 && taskNumber <= tasks.size()) {
    Task& task = tasks[taskNumber - 1];
    string name, description, dueDate;
    cout << "Enter new task name (current: "
        << task.getName() << "): ";
    cin.ignore();
    getline(cin, name);
    cout << "Enter new task description (current: "
        << task.getDescription() << "): ";
    getline(cin, description);
    cout << "Enter new task due date (current: "
        << task.getDueDate() << "): ";
    getline(cin, dueDate);

    task.setName(name);
    task.setDescription(description);
    task.setDueDate(dueDate);
}

```

```
        cout << "Task updated successfully!" << endl;
    }
    else {
        cout << "Invalid task number!" << endl;
    }
}
```

// Run the to-do list application

```
void run()
{
    int choice;
    do {
        displayMenu();
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                addTask();
                break;
            case 2:
                deleteTask();
                break;
            case 3:
                displayTasks();
                break;
            case 4:
                markTaskCompleted();
                break;
```



```
        case 5:
            editTask();
            break;
        case 6:
            cout << "Exiting program. Bye!" << endl;
            break;
        default:
            cout << "Invalid choice. Please try again!"
                << endl;
    }
} while (choice != 6);
}
};
```

// Main function

```
int main()
{
    // Create a ToDoList object and run the application
    ToDoList toDoList;
    toDoList.run();
    return 0;
}
```

❖ **OUTPUT:-**

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

----- To-Do List Menu -----

1. Add Task
2. Delete Task
3. Display Tasks
4. Mark Task as Completed
5. Edit Task
6. Exit

Enter your choice: 1

Enter task name: Cricket Bat Maintainance

Enter task description: add the grip to cricket bat handle

Enter task due date (YYYY-MM-DD): 2025-01-15

1. Add Task
2. Delete Task
3. Display Tasks
4. Mark Task as Completed
5. Edit Task
6. Exit

Enter your choice: 1

Enter task name: problem solving

Enter task description: solve POTD on the computer

Enter task due date (YYYY-MM-DD): 2025-01-03

Task added successfully!

----- To-Do List Menu -----

1. Add Task
2. Delete Task
3. Display Tasks
4. Mark Task as Completed
5. Edit Task
6. Exit

Enter your choice: 3

Tasks:

1. Cricket Bat Maintainance (Pending) - Due: 2025-01-15
Description: add the grip to cricket bat handle

```
2. problem solving (Pending) - Due: 2025-01-03
   Description: solve POTD on the computer
```

```
----- To-Do List Menu -----
```

1. Add Task
2. Delete Task
3. Display Tasks
4. Mark Task as Completed
5. Edit Task
6. Exit

```
-----
Enter your choice: 3
```

```
Enter your choice: 3
```

```
Tasks:
```

1. Cricket Bat Maintainance (Pending) - Due: 2025-01-15
 Description: add the grip to cricket bat handle
2. problem solving (Pending) - Due: 2025-01-03
 Description: solve POTD on the computer

```
----- To-Do List Menu -----
```

1. Add Task
2. Delete Task
3. Display Tasks
4. Mark Task as Completed

```
-----
Enter your choice: 2
```

```
Tasks:
```

1. Cricket Bat Maintainance
2. problem solving

```
Enter the task number to delete: 1
```

```
Task deleted successfully!
```

```
----- To-Do List Menu -----
```

1. Add Task
2. Delete Task
3. Display Tasks

3. Display Tasks
4. Mark Task as Completed
5. Edit Task
6. Exit

Enter your choice: 3

Tasks:

1. problem solving (Pending) - Due: 2025-01-03
Description: solve POTD on the computer

----- To-Do List Menu -----

1. Add Task