# TASK SCHEDULING SYSYTEM

Submitted by,
Robins Mathew
Roll No:63
Computer Science Department

## OVERVIEW OF PROJECT

This project is a Task Scheduling System designed to help users efficiently manage their tasks by adding, removing, displaying, and marking tasks as complete. The problem it addresses is the challenge of organizing and prioritizing multiple tasks in a busy schedule. Each task has an associated name, priority level, and estimated hours to complete, which are automatically assigned based on priority. The system maintains a list of tasks, allowing users to view all tasks along with their details, and update the status of tasks as they are completed. The objective is to provide an easy-to-use tool that enhances productivity by simplifying task management and ensuring that high-priority tasks are given appropriate attention.

## PROBLEM STATEMENT

Managing multiple tasks efficiently can be challenging, especially when dealing with varying priorities and deadlines. Without a proper system, it is easy to lose track of important tasks, leading to missed deadlines and decreased productivity. The lack of organization and prioritization can result in an overwhelming workload, making it difficult to focus on critical tasks.

## OBJECTIVE

The objective of this Task Scheduling System is to provide users with a simple and effective tool for managing their tasks. The system allows users to add, remove, display, and mark tasks as complete. By assigning hours based on priority levels, the system helps users prioritize their workload, ensuring that high-priority tasks are given the appropriate attention. This enhances productivity and ensures that tasks are completed efficiently and on time.

## Minimum Software and Hardware Requirements

Software Requirements

1. Operating System: Windows 10 or higher, macOS, or a Linux distribution
2. Compiler: GCC (GNU Compiler Collection) or any standard C compiler
3. IDE/Text Editor: Code::Blocks, Visual Studio Code, or any preferred text editor
4. Standard C Library: Included with the compiler

Hardware Requirements

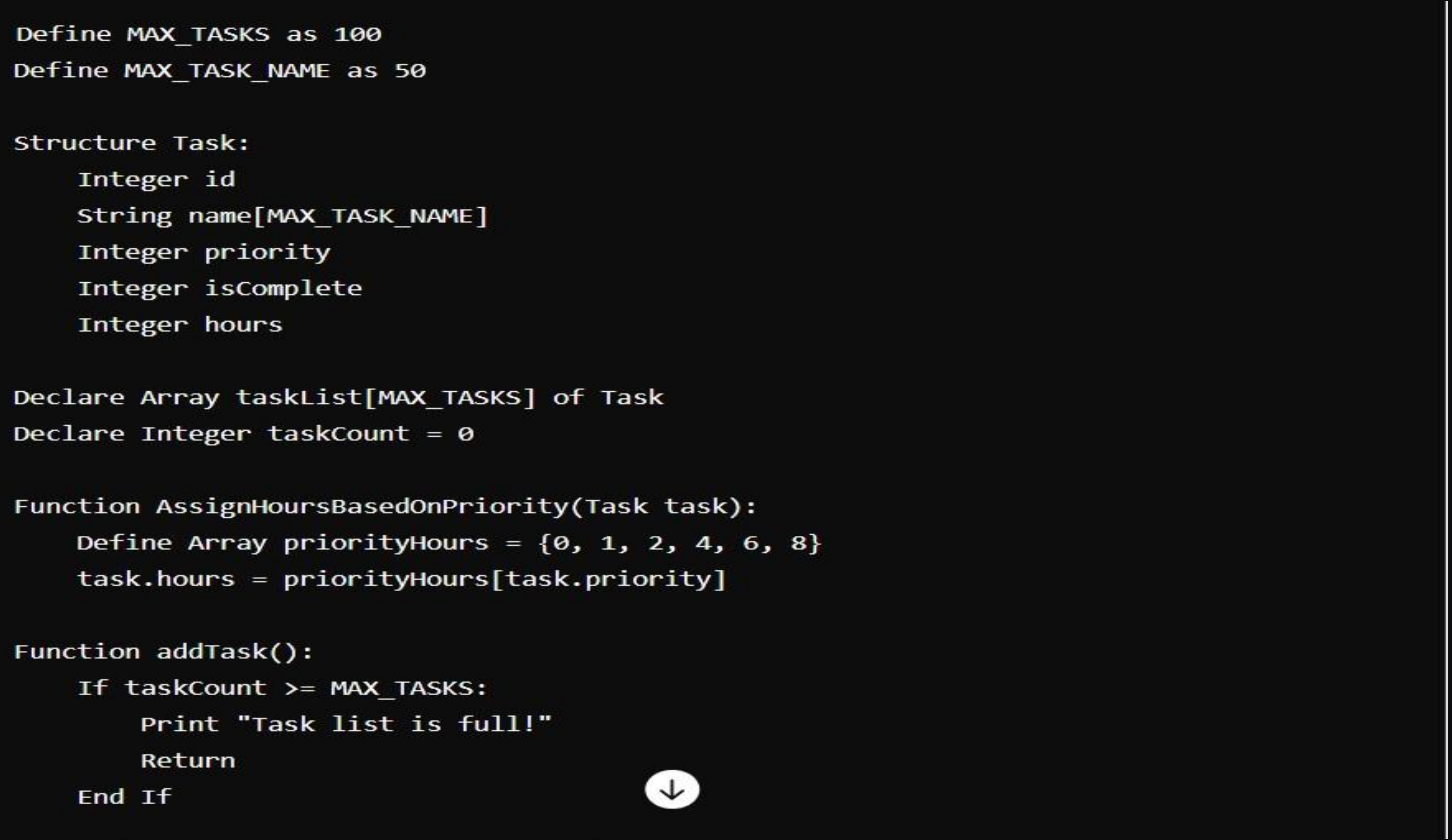
1. Processor: 1 GHz or faster processor
2. RAM: 1 GB or more
3. Storage: 100 MB of available disk space
4. Display: 800x600 resolution or higher
5. Input Devices: Keyboard and mouse

## DESIGN AND DEVELOPMENT

### PROGRAM LOGIC

The Task Scheduling System is a simple C program designed to manage tasks efficiently.It uses a Task structure to store details like ID, name, priority, completion status, and estimated hours. The program maintains a global array of tasks and a count of the tasks. Key functions include adding tasks with user input, removing tasks by ID, displaying all tasks, and marking tasks as complete. The Assign HoursBased On Priority function assigns estimated hours based on task priority. A menu function provides options for these operations, enabling easy interaction. The main function initiates the program, running an infinite loop until the user chooses to exit. This design ensures an organized approach to task management, enhancing productivity by prioritizing tasks effectively.

## PSEUDO CODE



```
Define MAX_TASKS as 100
Define MAX_TASK_NAME as 50

Structure Task:
    Integer id
    String name[MAX_TASK_NAME]
    Integer priority
    Integer isComplete
    Integer hours

Declare Array taskList[MAX_TASKS] of Task
Declare Integer taskCount = 0

Function AssignHoursBasedOnPriority(Task task):
    Define Array priorityHours = {0, 1, 2, 4, 6, 8}
    task.hours = priorityHours[task.priority]

Function addTask():
    If taskCount >= MAX_TASKS:
        Print "Task list is full!"
        Return
    End If
    Declare Task newTask
    newTask.id = taskCount + 1
    newTask.isComplete = 0

    Print "Enter task name: "
    Read newTask.name

    Print "Enter task priority (1-5): "
    Read newTask.priority

    Call AssignHoursBasedOnPriority(newTask)

    taskList[taskCount] = newTask
    taskCount = taskCount + 1

    Print "Task added successfully!"

Function removeTask():
    If taskCount == 0:
        Print "No tasks to remove!"
        Return
    End If
    Print "Enter task ID to remove: "
    Read Integer id

    For i from 0 to taskCount - 1:
        If taskList[i].id == id:
            For j from i to taskCount - 2:
                taskList[j] = taskList[j + 1]
            End For
            taskCount = taskCount - 1
            Print "Task removed successfully!"
            Return
        End If
    End For

    Print "Task with ID " id " not found!"

Function displayTasks():
    If taskCount == 0:
        Print "No tasks to display!"
        Return
    End If
    End For

    Print "Task with ID " id " not found!"

Function displayTasks():
    If taskCount == 0:
        Print "No tasks to display!"
        Return
    End If

    Print "ID\tName\t\tPriority\tHours\tStatus"
    For i from 0 to taskCount - 1:
        Print taskList[i].id "\t" taskList[i].name "\t\t" taskList[i].priority "\t\t" task

Function markTaskComplete():
    If taskCount == 0:
        Print "No tasks to mark as complete!"
        Return
    End If

    Print "Enter task ID to mark as complete: "
    Read Integer id
```

```
Function markTaskComplete():
    If taskCount == 0:
        Print "No tasks to mark as complete!"
        Return
    End If

    Print "Enter task ID to mark as complete: "
    Read Integer id

    For i from 0 to taskCount - 1:
        If taskList[i].id == id:
            taskList[i].isComplete = 1
            Print "Task marked as complete!"
            Return
        End If
    End For

    Print "Task with ID " id " not found!"
```

```
Function menu():
    While True:
        Print "\nTask Management System"
        Print "1. Add Task"
        Print "2. Remove Task"
        Print "3. Display Tasks"
        Print "4. Mark Task as Complete"
        Print "5. Exit"
        Print "Enter your choice: "

        Read Integer choice

        Switch choice:
            Case 1:
                Call addTask()
            Case 2:
                Call removeTask()
            Case 3:
                Call displayTasks()
            Case 4:
                Call markTaskComplete()
            Case 5:
                Exit Program
                Call addTask()
            Case 2:
                Call removeTask()
            Case 3:
                Call displayTasks()
            Case 4:
                Call markTaskComplete()
            Case 5:
                Exit Program
            Default:
                Print "Invalid choice!"

Function main():
    Call menu()
```

## TEST CASES AND RESULTS

### TEST CASES

**1.Add Task**

1. **Input**: Task Name: "Task A", Priority: 3
2. **Expected Output**: Task added successfully with ID: 1, Hours: 4

**2.Remove Task**

1. **Input**: Task ID: 1
2. **Expected Output**: Task removed successfully

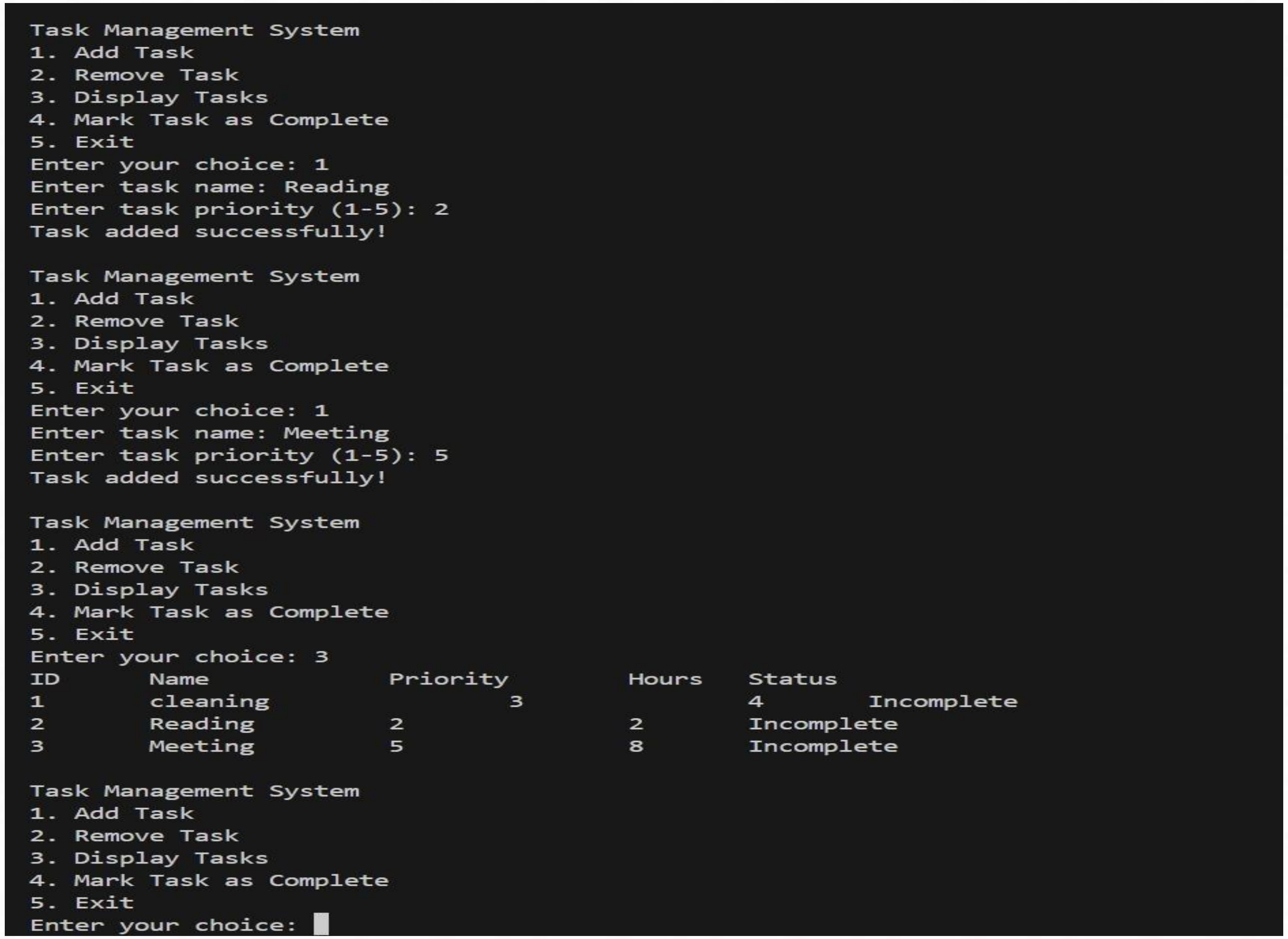
**3.Display Tasks**

1. **Input**: None
2. **Expected Output**: Displays all tasks with their details (ID, Name, Priority, Hours, Status)

**4.Mark Task as Complete**

1. **Input**: Task ID: 1
2. **Expected Output**: Task marked as complete

## OUTPUT



```
Task Management System
1. Add Task
2. Remove Task
3. Display Tasks
4. Mark Task as Complete
5. Exit
Enter your choice: 1
Enter task name: Reading
Enter task priority (1-5): 2
Task added successfully!

Task Management System
1. Add Task
2. Remove Task
3. Display Tasks
4. Mark Task as Complete
5. Exit
Enter your choice: 1
Enter task name: Meeting
Enter task priority (1-5): 5
Task added successfully!

Task Management System
1. Add Task
2. Remove Task
3. Display Tasks
4. Mark Task as Complete
5. Exit
Enter your choice: 3
ID      Name            Priority        Hours   Status
1       cleaning                3               4       Incomplete
2       Reading         2               2       Incomplete
3       Meeting         5               8       Incomplete

Task Management System
1. Add Task
2. Remove Task
3. Display Tasks
4. Mark Task as Complete
5. Exit
Enter your choice: 
```

## Discussion of Results

The Task Scheduling System was tested with various scenarios, including adding, removing, displaying, and marking tasks as complete. The system successfully added tasks, correctly assigned hours based on priority, and displayed the task details accurately. Tasks were also removed and marked as complete as expected. The results confirmed that the system functions correctly, providing an effective tool for task management. The implementation met the objectives of organizing tasks and prioritizing them efficiently.

## Summary

The Task Scheduling System is a straightforward C program designed to manage tasks efficiently. It allows users to add tasks with priorities, assign estimated hours based on priority, display task details, mark tasks as complete, and remove tasks. The program maintains a list of tasks and provides a simple menu-driven interface for interacting with the task list. Through testing, the system demonstrated its ability to handle basic task management operations effectively, meeting its objective of organizing and prioritizing tasks.

## FUTURE ENHANCEMENTS

1. Save and load task: Add file handling to save tasks and load them when the program starts.
2. Edit Tasks: Allow users to modify existing task details.
3. User Accounts: Implement user login to manage personal task lists.
4. Due Dates: Add deadlines and reminders for tasks.
5. Task Categories: Enable task categorization for better organization.
6. Graphical Interface: Create a GUI for easier interaction.
7. Search and Filter: Implement features to search and filter tasks.

Thank You!