Name: Atharva Maithani
SAP-ID: 590024379
Batch-19

C PROJECT SMART TASK MANAGER

INDEX

TOPIC	PAGE NUMBER
ABSTRACT	1
PROBLEM DEFINITION	2
SYSTEM DESIGN	3-4
IMPLEMENTATION DETAILS	5-7
TESTING AND RESULTS	8
CONCLUSION AND FUTURE WORK	9
REFERENCES	10

ABSTRACT

Task management plays a central role in increasing productivity, structuring workflow, and fulfilling one's responsibilities well in advance. At both academic and professional levels, many often face a problem regarding selecting the right priority of tasks, which further delays work and results in inefficiency and poor performance. This project, Smart Task Manager, proposes a lightweight, command-line based task management system to handle such problems with structured task organization and intelligent features of priority decay and productivity scoring. This system is entirely implemented in the C programming language; it represents the practical usage of core programming concepts along with a functioning tool for improving day-to-day task handling.

The system's main goal is to give users the opportunity to create, view, track, and finish tasks intuitively and easily. Users will be able to add tasks with such details as title, description, priority value, and estimated completion time. Unlike other static lists of tasks, the Smart Task Manager introduces a dynamic mechanism of priority decay: each time the user views the list of tasks, pending tasks have their priority reduced. This recreates in the model a real-life consequence of delaying responsibilities and encourages timely action, making the system more realistic and behavior-driven compared to ordinary task managers.

Another important feature inculcated into the system is the productivity score, measured whenever a user completes his or her task. The score rises according to the priority and estimated time of the completed task, thus giving a simple gamified element that encourages users to pay more attention to the important tasks. This score system not only helps in measuring productivity but also gives the user feedback on his work patterns; therefore, enabling him to reflect on his efficiency.

The system is implemented based on the essential elements of C programming, including structures, arrays, conditional statements, loops, and the handling of user input using functions like fgets() and sscanf() for reliability. In the end, the Smart Task Manager fulfills its objective of implementing an effective, interactive, and logically structured task management system in addition to the illustration of some fundamental concepts of the C language. This project not only fulfills academic requirements but gives a meaningful tool that can be expanded upon for real-world use.

PROBLEM DEFINITION

Effective task management helps people increase productivity, organize their duties, and meet time limits. However, most classic methods of task management, such as handwritten lists, basic mobile notes, or static digital checklists, fail to handle the dynamic nature of how tasks evolve with time. Users frequently experience problems like forgetting, misjudging the importance of some particular tasks, procrastinating due to unclear priorities, or a lack of motivation to do more demanding tasks. Such limitations indicate the need for a system that will store the tasks but also react smartly to user behavior and changing priorities.

The objective of the project entitled Smart Task Manager Using C is to design and implement a simple, interactive, and dynamic command-line tool that addresses these general issues within a structured programming environment. This system is envisioned to enable users to add tasks, view pending tasks, keep track of completion, and ultimately monitor productivity efficiently. One problem this project addresses that is not found in many of the simpler task lists is the lack of adaptive prioritization. The system will solve this issue by using a priority decay mechanism where tasks automatically lose priority when repeatedly viewed and left incomplete. This simulates real-world consequences of delay and encourages timely decision-making.

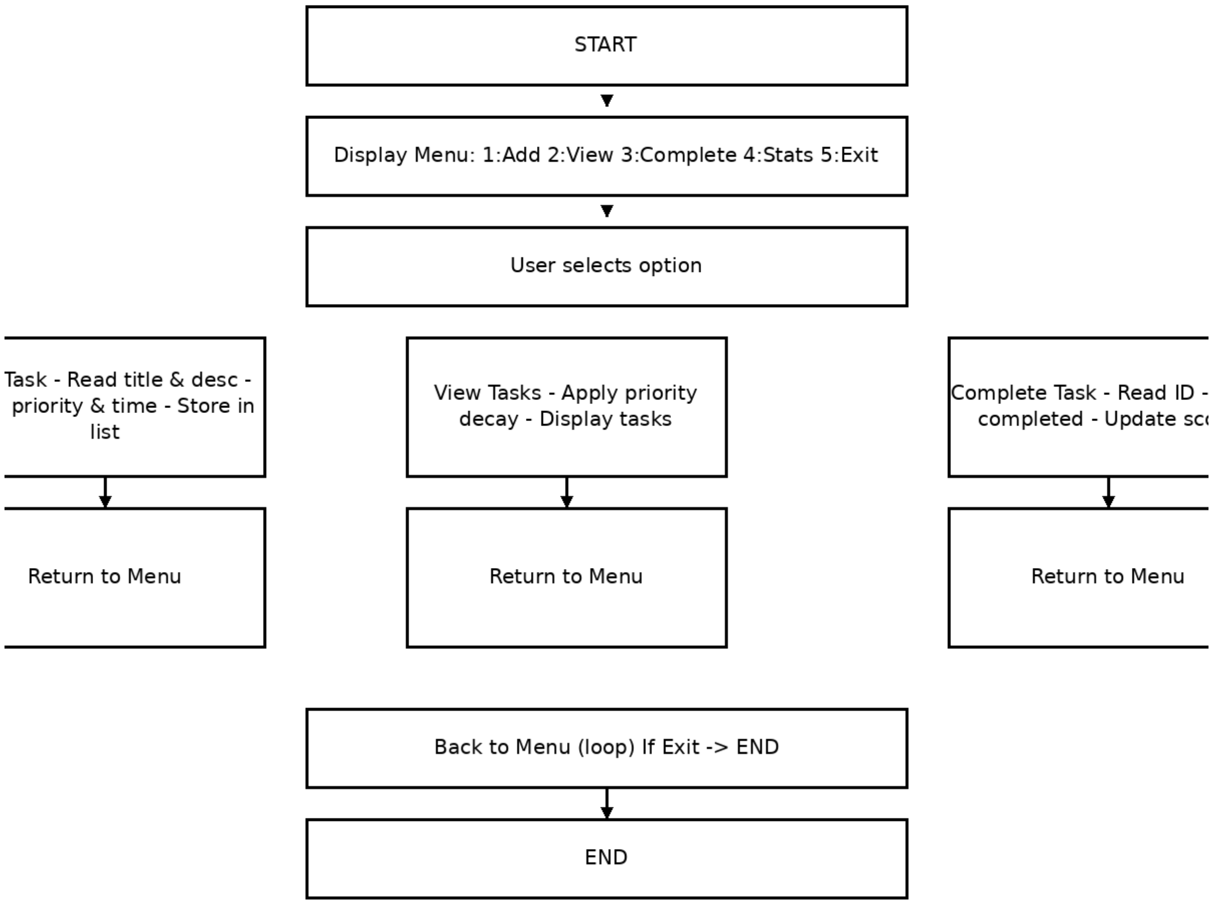
Another challenge that it tries to address is the absence of feedback on the part of users for their productivity and task completion patterns. It introduces a productivity score into the system that would keep increasing by the priority and estimated effort of the tasks completed. This scoring mechanism offers motivational feedback to users to help them establish better task-handling habits.

Also, many users, particularly beginners in the field of programming, need a system that is simple, lightweight, and free from unnecessary complexity. The system is thus intentionally designed without file storage, highly accessible and easy to understand, while still being able to demonstrate core C language concepts such as structures, arrays, loops, conditionals, and menu-driven logic.

In general, the problem to be solved is the necessity of having an efficient and intelligent task management tool that enhances productivity, but simple enough for its implementation and understanding within the scope of academic programming.

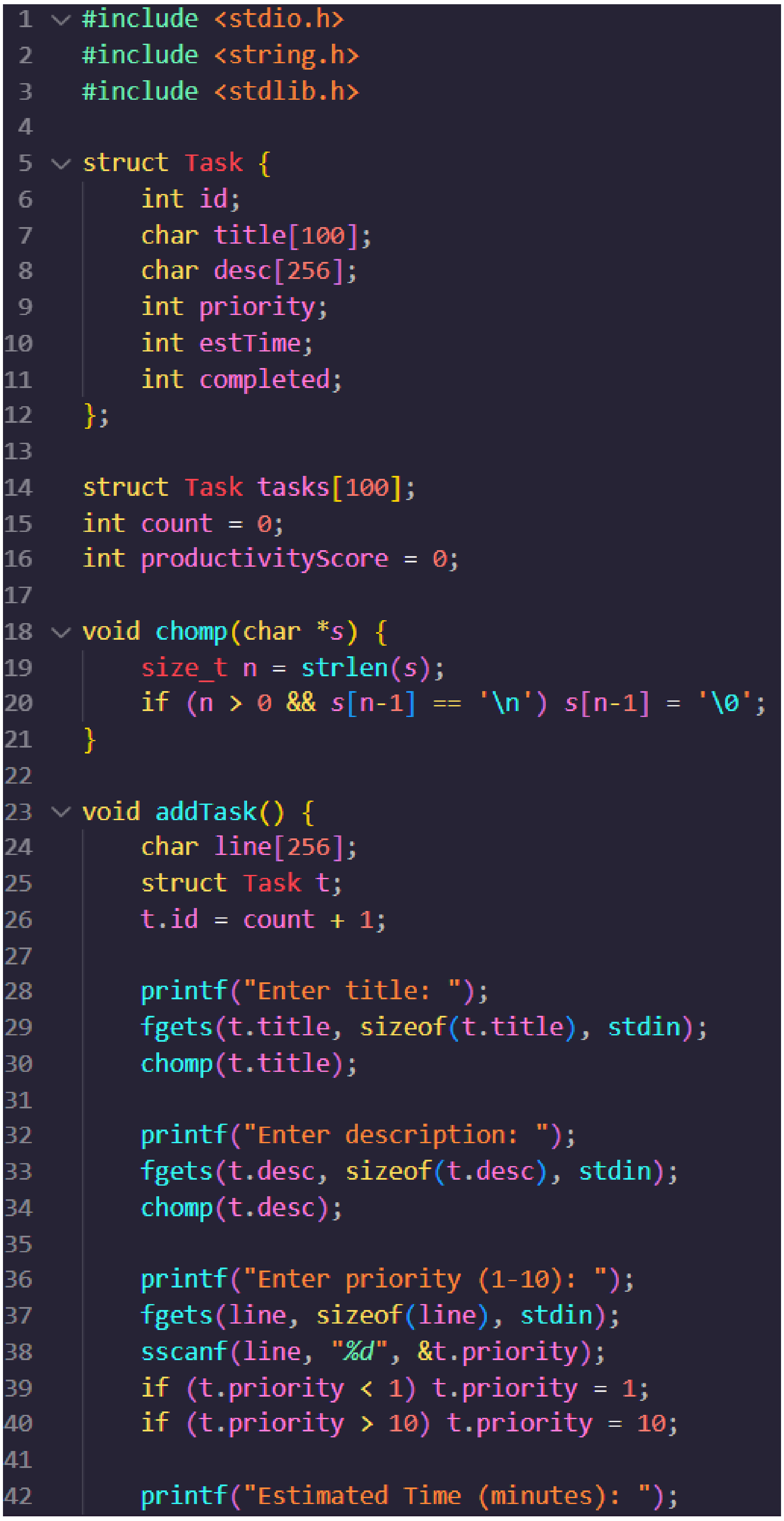
SYSTEM DESIGN

Flowchart: Smart Task Manager

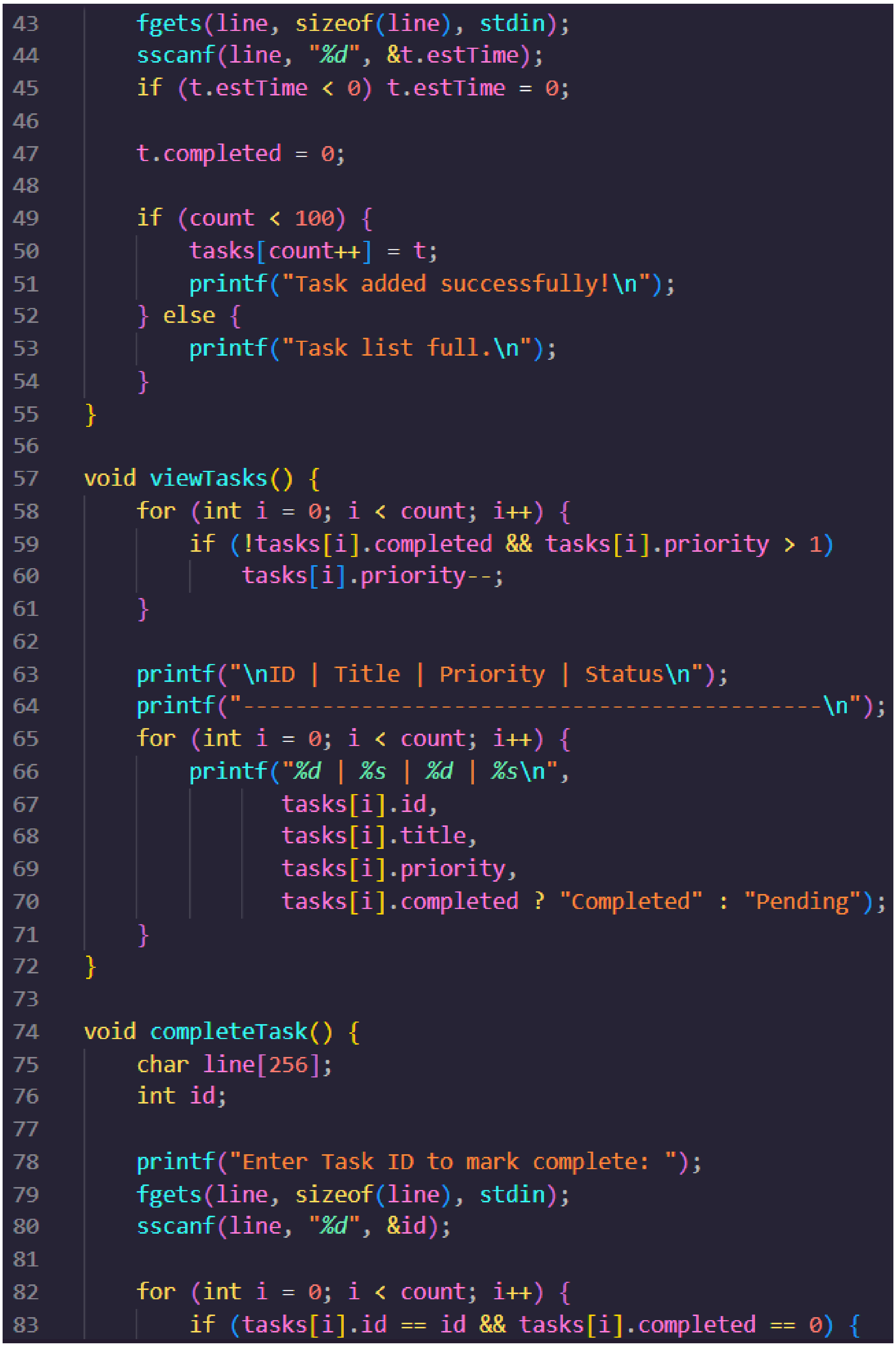

START
Display Menu: 1:Add 2:View 3:Complete 4:Stats 5:Exit
User selects option
Task - Read title & desc - priority & time - Store in list
View Tasks - Apply priority decay - Display tasks
Complete Task - Read ID - completed - Update sc
Return to Menu
Return to Menu
Return to Menu
Back to Menu (loop) If Exit -> END
END

```
START
Initialize task list
Set count = 0
Set productivityScore = 0
REPEAT
    Display menu:
        1. Add Task
        2. View Tasks
        3. Complete Task
        4. Show Statistics
        5. Exit
    Read user choice
    IF choice = 1 THEN
        Read task details
        Store task in list
        Increase count
    ENDIF
    IF choice = 2 THEN
        Apply priority decay to pending tasks
        Display all tasks
    ENDIF
    IF choice = 3 THEN
        Read task ID
        Mark task as completed
        Update productivityScore
    ENDIF

    IF choice = 4 THEN
        Display total tasks, pending, completed, and productivityScore
    ENDIF
UNTIL choice = 5
END
```

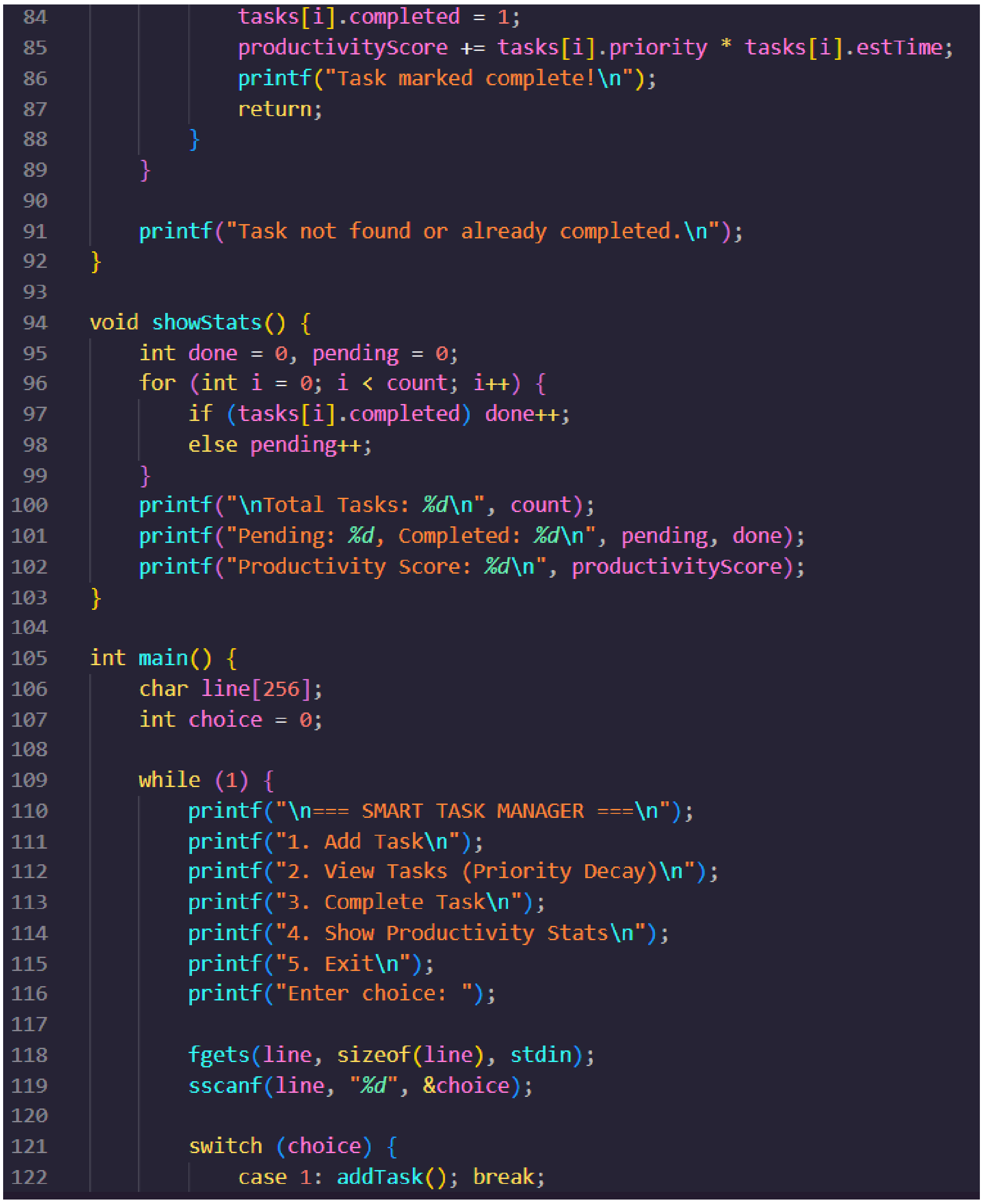
IMPLEMENTATION DETAILS



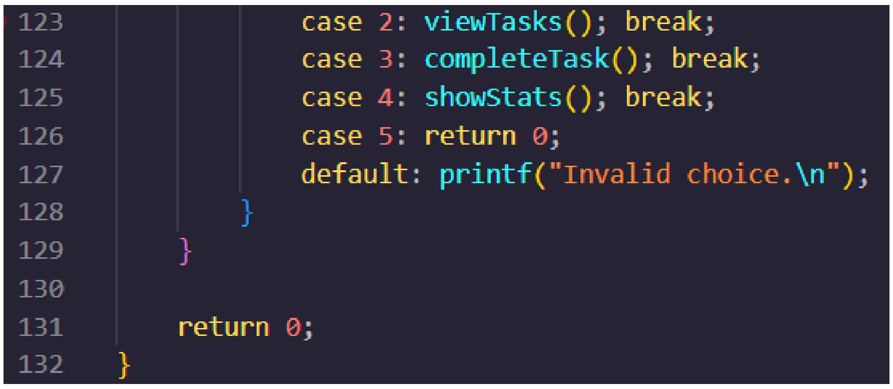
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct Task {
    int id;
    char title[100];
    char desc[256];
    int priority;
    int estTime;
    int completed;
};

struct Task tasks[100];
int count = 0;
int productivityScore = 0;

void chomp(char *s) {
    size_t n = strlen(s);
    if (n > 0 && s[n-1] == '\n') s[n-1] = '\0';
}

void addTask() {
    char line[256];
    struct Task t;
    t.id = count + 1;

    printf("Enter title: ");
    fgets(t.title, sizeof(t.title), stdin);
    chomp(t.title);

    printf("Enter description: ");
    fgets(t.desc, sizeof(t.desc), stdin);
    chomp(t.desc);

    printf("Enter priority (1-10): ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%d", &t.priority);
    if (t.priority < 1) t.priority = 1;
    if (t.priority > 10) t.priority = 10;

    printf("Estimated Time (minutes): ");
```



```c
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%d", &t.estTime);
    if (t.estTime < 0) t.estTime = 0;

    t.completed = 0;

    if (count < 100) {
        tasks[count++] = t;
        printf("Task added successfully!\n");
    } else {
        printf("Task list full.\n");
    }
}

void viewTasks() {
    for (int i = 0; i < count; i++) {
        if (!tasks[i].completed && tasks[i].priority > 1)
            tasks[i].priority--;
    }

    printf("\nID | Title | Priority | Status\n");
    printf("--------------------------------------------\n");
    for (int i = 0; i < count; i++) {
        printf("%d | %s | %d | %s\n",
               tasks[i].id,
               tasks[i].title,
               tasks[i].priority,
               tasks[i].completed ? "Completed" : "Pending");
    }
}

void completeTask() {
    char line[256];
    int id;

    printf("Enter Task ID to mark complete: ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%d", &id);

    for (int i = 0; i < count; i++) {
        if (tasks[i].id == id && tasks[i].completed == 0) {
```

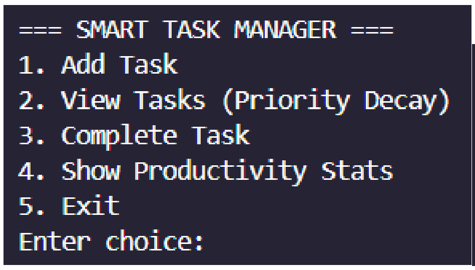


```c
            tasks[i].completed = 1;
            productivityScore += tasks[i].priority * tasks[i].estTime;
            printf("Task marked complete!\n");
            return;
        }
    }

    printf("Task not found or already completed.\n");
}

void showStats() {
    int done = 0, pending = 0;
    for (int i = 0; i < count; i++) {
        if (tasks[i].completed) done++;
        else pending++;
    }
    printf("\nTotal Tasks: %d\n", count);
    printf("Pending: %d, Completed: %d\n", pending, done);
    printf("Productivity Score: %d\n", productivityScore);
}

int main() {
    char line[256];
    int choice = 0;

    while (1) {
        printf("\n=== SMART TASK MANAGER ===\n");
        printf("1. Add Task\n");
        printf("2. View Tasks (Priority Decay)\n");
        printf("3. Complete Task\n");
        printf("4. Show Productivity Stats\n");
        printf("5. Exit\n");
        printf("Enter choice: ");

        fgets(line, sizeof(line), stdin);
        sscanf(line, "%d", &choice);

        switch (choice) {
            case 1: addTask(); break;
```

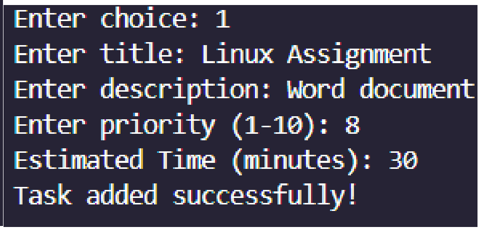


```c
            case 2: viewTasks(); break;
            case 3: completeTask(); break;
            case 4: showStats(); break;
            case 5: return 0;
            default: printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

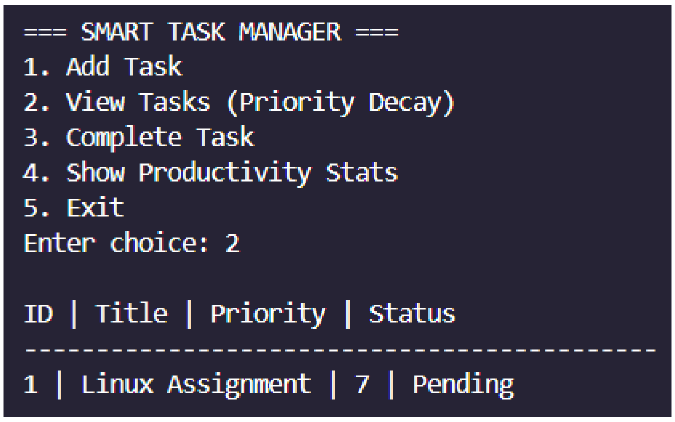
TESTING AND RESULTS



```
=== SMART TASK MANAGER ===
1. Add Task
2. View Tasks (Priority Decay)
3. Complete Task
4. Show Productivity Stats
5. Exit
Enter choice:
```



```
Enter choice: 1
Enter title: Linux Assignment
Enter description: Word document
Enter priority (1-10): 8
Estimated Time (minutes): 30
Task added successfully!
```



```
=== SMART TASK MANAGER ===
1. Add Task
2. View Tasks (Priority Decay)
3. Complete Task
4. Show Productivity Stats
5. Exit
Enter choice: 2

ID | Title | Priority | Status
--------------------------------------------
1 | Linux Assignment | 7 | Pending
```

As seen above, I created a task called ‘Linux Assignment’ by adding all the required details. The program also stores the list of tasks in an organized manner.

CONCLUSION AND FUTURE WORK

The Smart Task Manager successfully meets its objective in coming up with a simple, efficient, and interactive task-handling system that incorporates the core concepts of structured programming. The project offers a functional tool that simulates real-world challenges in managing tasks through features such as adding tasks, marking tasks as completed, viewing pending tasks with automatic priority decay, and maintaining a productivity score. In spite of the fact that this console-based program does not have any external storage, the system clearly illustrates how logical design and modular programming can produce meaningful, usable applications.

One of the key strengths of the system is its inclusion of dynamic priority decay. Unique among typical student-level task managers, this behavioral feature entails tasks losing priority when frequently viewed but left uncompleted, thereby encouraging users to make quicker decisions and avoid procrastination. Furthermore, the productivity score mechanism provides its own kind of motivational feedback when high-priority or time-intensive tasks are completed. Together, these features highlight the potential for simple algorithms to enhance user engagement and efficiency.

While the program works well within its current scope, it also provides a number of avenues for extension. File handling in a future edition would provide the ability to save and retrieve tasks between runs of the program, thus making the utility much more usable in everyday life. Other possible extensions include, but are not limited to, sorting tasks by priority or deadline, adding categories or labels, colorizing terminal output for readability, and implementing deadline reminders. More advanced extensions could be to implement a GUI, or even port the system onto a mobile or web platform. These extensions would offer broader usability and functionality for the project and further opportunities to apply more advanced aspects of programming.

The final project is well-developed and includes much potential for expansion, learning, and the ability to be used in the real world.

REFERENCES

https://www.geeksforgeeks.org

https://www.youtube.com/@BroCodez