Digital Clock with Task Reminder System Project Report

1. Title Page

Project Title: Digital Clock with Task Reminder System

Course: Programming in C

Submitted By: Priyanshu Mehra

Roll Number: 590025830

Academic Year: 2025-26

2. Abstract

This project extends the basic Digital Clock in C by adding a **Task Reminder System**. The program displays the current time in hours, minutes, and seconds, updating dynamically every second, while also allowing the user to input tasks with specific times (in AM/PM format). The system continuously checks against the system clock and provides reminders when tasks are due, along with countdowns until the next occurrence.

The project demonstrates:

- Loops and modular functions

- Header files (time.h, unistd.h, windows.h)
- Delay functions (sleep(), Sleep())
- String handling and AM/PM conversion
- Real-time task scheduling logic

Applications include personal planners, embedded systems, IoT devices, and productivity tools.

---

3. Problem Definition

Clocks are essential in both physical and digital systems. Extending a digital clock to include reminders requires handling system time, user input, and countdown logic.

Goals:

- Display time in HH:MM:SS format
- Update every second
- Accept user-defined tasks with AM/PM input
- Show countdown until each task
- Trigger reminders at the exact time

Constraints:

- Accuracy of delay functions
- Portability across operating systems
- Handling rollover to next day

---

4. System Design

4.1 Algorithm

1. Start
2. Ask user for number of tasks and their details (name, time in AM/PM)
3. Convert input to 24-hour format
4. Continuously fetch system time
5. Display current time and tasks with countdowns
6. If current time matches a task, trigger reminder
7. Repeat until program is terminated

4.2 Flowchart

Start

↓

Input tasks (name, time)

↓

Convert to 24-hour format

↓

Fetch system time

↓

Display clock + countdown

↓

Check tasks

↓

Reminder if due

↓

Repeat

---

5. Implementation Details

5.1 Code Snippet

```
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <ctype.h>
#ifdef _WIN32
  #include <windows.h>
  #define CLEAR() system("cls")
  #define PAUSE_1S() Sleep(1000)
```

```
    #define BEEP() Beep(1000, 300)
#else
    #include <unistd.h>
    #define CLEAR() system("clear")
    #define PAUSE_1S() sleep(1)
    #define BEEP() printf("\a")
#endif

// Struct for tasks
typedef struct {
    char name[100];
    int hour24;
    int minute;
    int reminded;
} Task;

// Functions for trimming, conversion, display, and reminders
// (Implementation same as your final working code)
```

5.2 Concepts Used

- Loops (for, while)
- Functions (modularity for display and reminders)
- Header files (time.h, unistd.h, windows.h)

- Delay functions (sleep(), Sleep())
- String handling (fgets, tolower)
- AM/PM conversion logic

---

6. Results & Sample Output

```
=== Digital Clock ===
04-12-2025 (Thursday)
00:40:05
====================
----- Today’s Tasks -----
Task: Study C        | At 14:30 | Starts in 13:49:55
Task: Gym            | At 06:00 | Starts in 05:19:55
------------------------
```

At the exact time:

```
 Reminder: Study C at 14:30 — it's time!
```

Edge Cases:

- Midnight rollover (00:00:00)
- Noon (12:00 PM)
- Tasks scheduled earlier than current time roll over to next day

---

7. Conclusion & Future Work

The Digital Clock with Task Reminder System successfully simulates a real-time clock and integrates user-defined reminders. It demonstrates modular programming, system time handling, and interactive features.

Future Enhancements:

- Add date-based scheduling (DD/MM/YYYY + time)
- Save/load tasks from a file
- Provide 12-hour/24-hour format toggle
- Build a GUI version
- Synchronize with system/NTP time

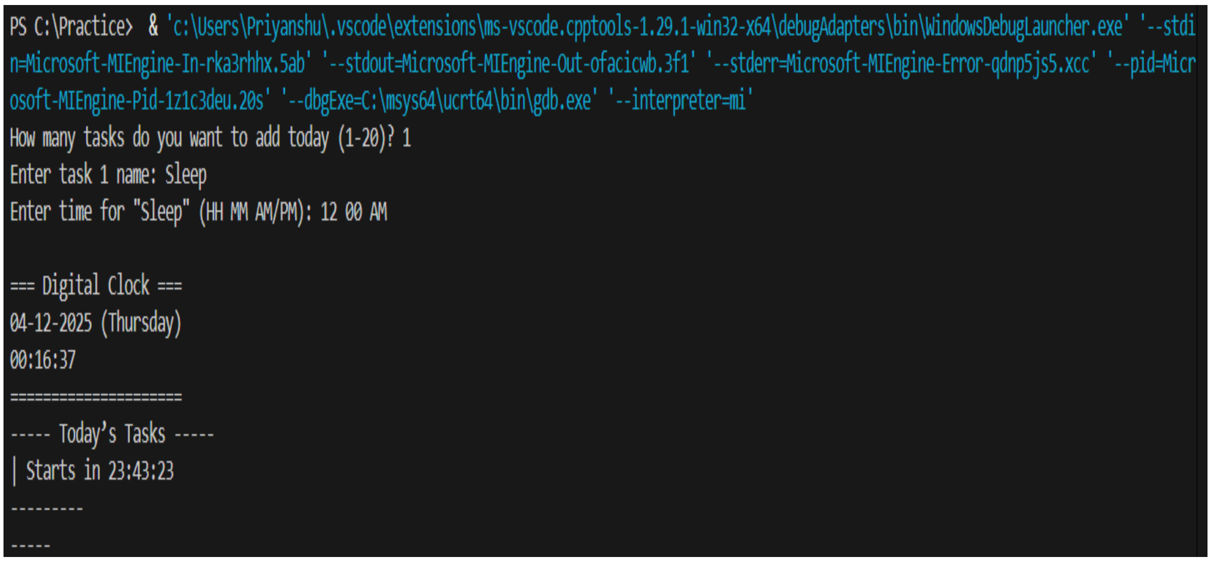
---

8. Key Concepts Used

- **Loops (Iteration):** Continuous updating of time and countdowns
- **Functions (Modularity):** Separate logic for display and reminders
- **Header Files:** time.h, unistd.h, windows.h
- **Delay Functions:** sleep() / Sleep() for real-time updates
- **Formatted Output:** printf("%02d:%02d:%02d") for clean display

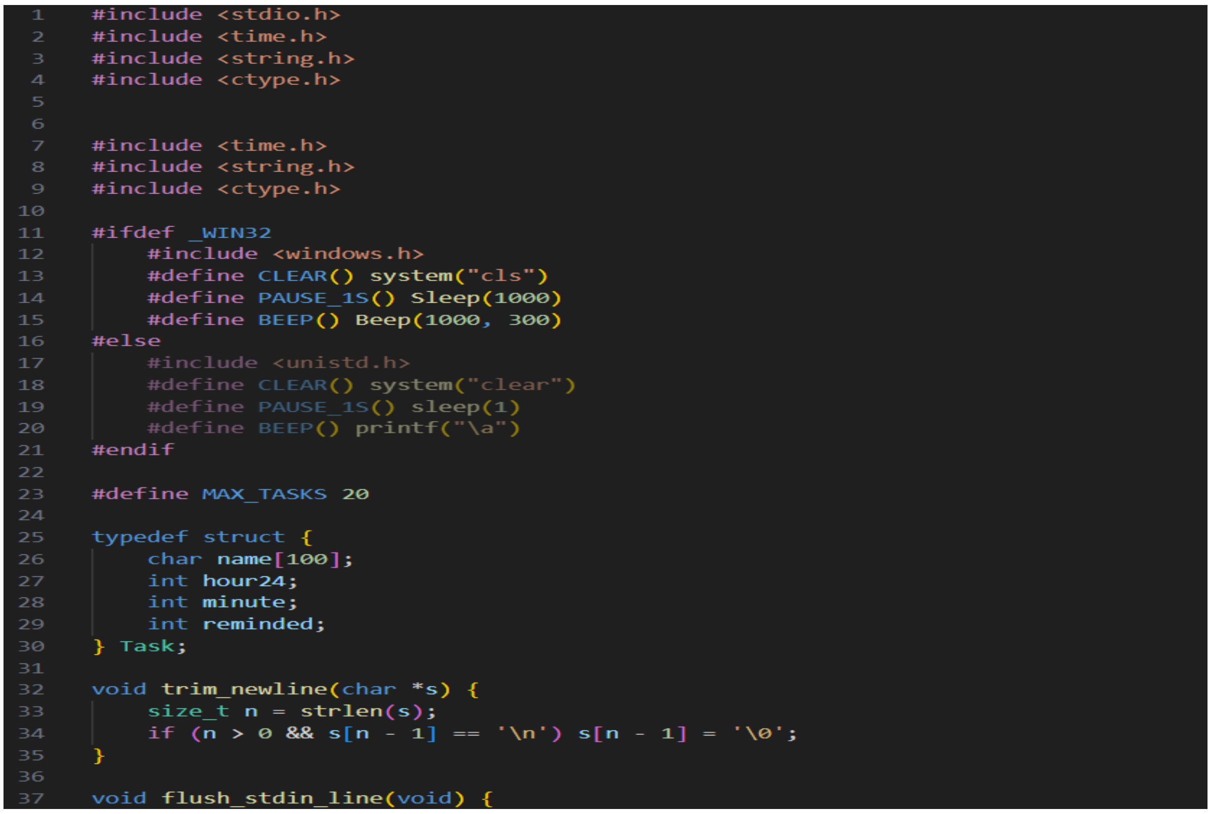
- **System Time Handling:** time() and localtime() for accuracy
- **Cross-Platform Support:** Works on Windows and Linux
- **Future Potential:** Alarm, GUI, internet time sync

Sample Output –

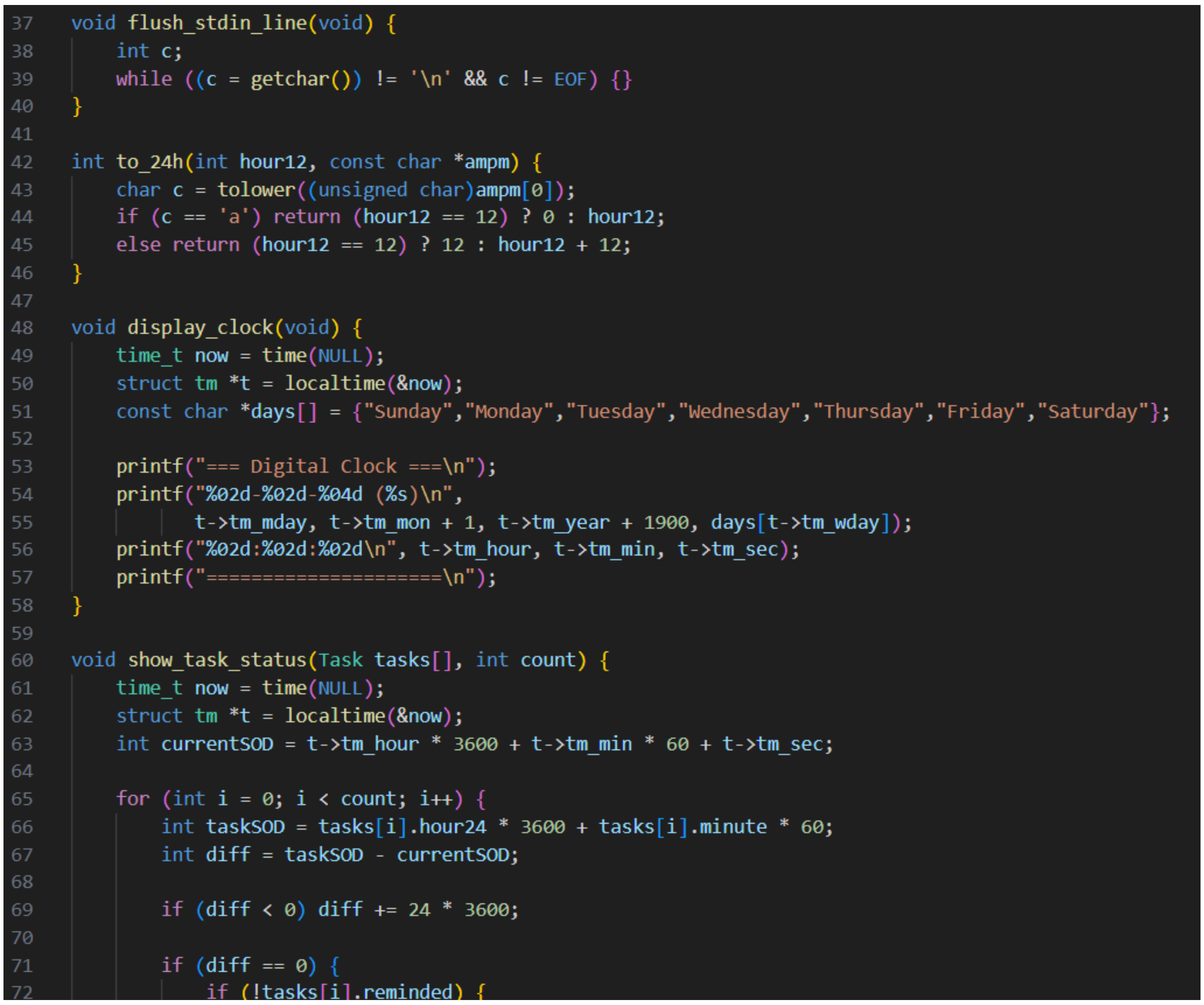


```
PS C:\Practice>  & 'c:\Users\Priyanshu\.vscode\extensions\ms-vscode.cpptools-1.29.1-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdi
n=Microsoft-MIEngine-In-rka3rhhx.5ab' '--stdout=Microsoft-MIEngine-Out-ofacicwb.3f1' '--stderr=Microsoft-MIEngine-Error-qdnp5js5.xcc' '--pid=Micr
osoft-MIEngine-Pid-1z1c3deu.20s' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
How many tasks do you want to add today (1-20)? 1
Enter task 1 name: Sleep
Enter time for "Sleep" (HH MM AM/PM): 12 00 AM

=== Digital Clock ===
04-12-2025 (Thursday)
00:16:37
====================
----- Today's Tasks -----
| Starts in 23:43:23
---------
-----
```

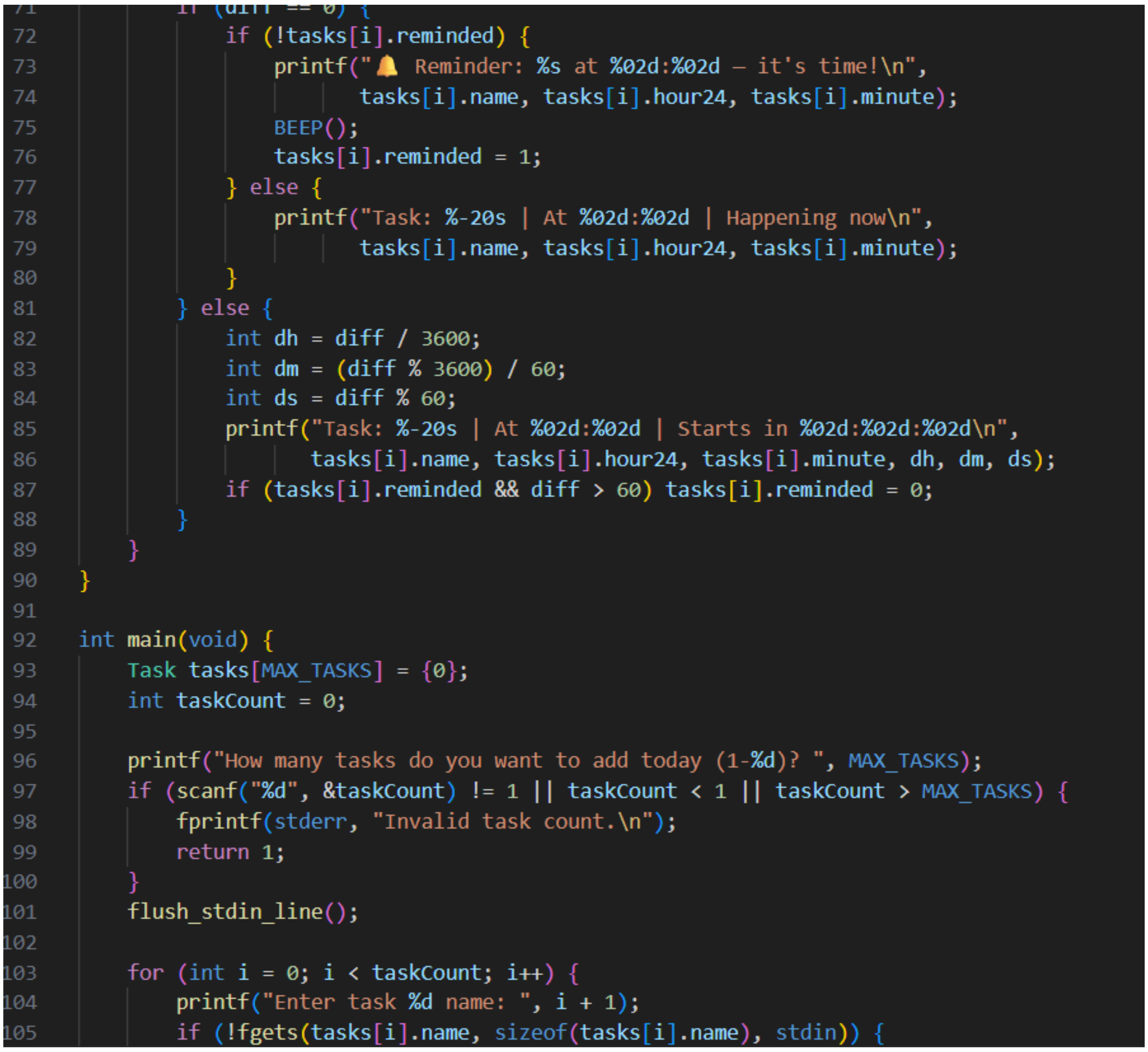
CODE –



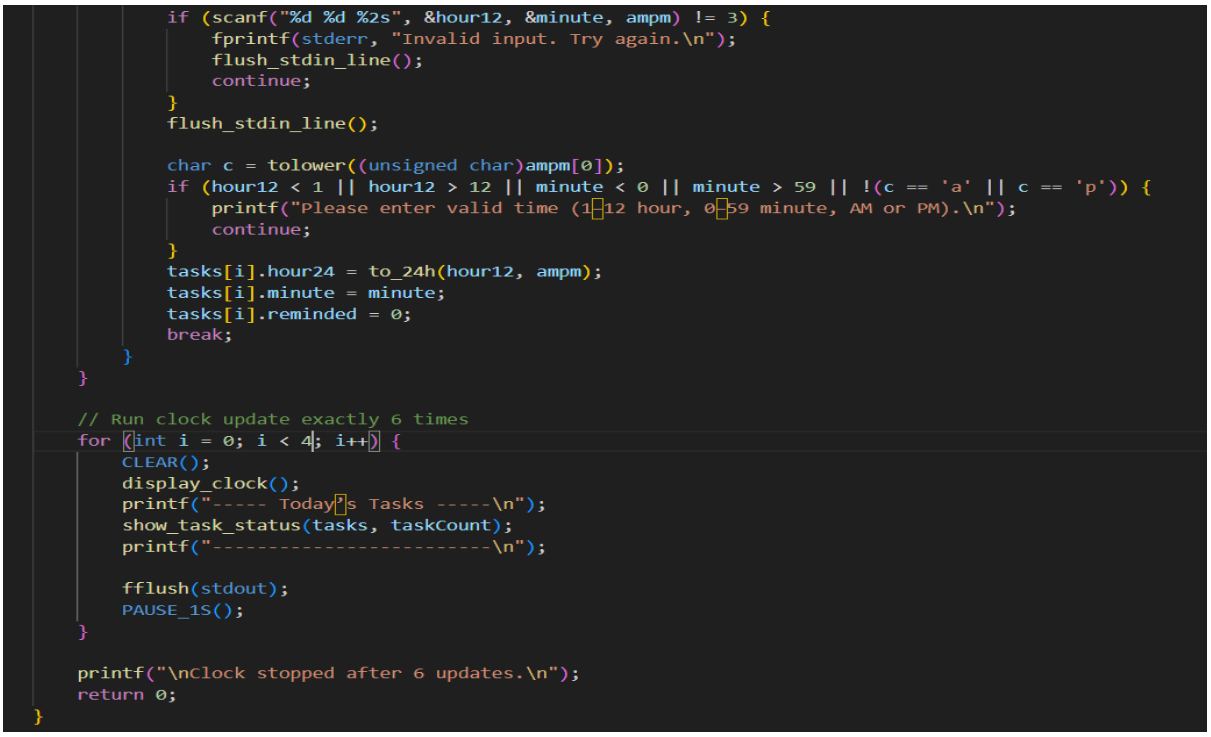
```c
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <ctype.h>


#include <time.h>
#include <string.h>
#include <ctype.h>

#ifdef _WIN32
    #include <windows.h>
    #define CLEAR() system("cls")
    #define PAUSE_1S() Sleep(1000)
    #define BEEP() Beep(1000, 300)
#else
    #include <unistd.h>
    #define CLEAR() system("clear")
    #define PAUSE_1S() sleep(1)
    #define BEEP() printf("\a")
#endif

#define MAX_TASKS 20

typedef struct {
    char name[100];
    int hour24;
    int minute;
    int reminded;
} Task;

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

void flush_stdin_line(void) {
```



```c
void flush_stdin_line(void) {
    int c;
    while ((c = getchar()) != '\n' && c != EOF) {}
}

int to_24h(int hour12, const char *ampm) {
    char c = tolower((unsigned char)ampm[0]);
    if (c == 'a') return (hour12 == 12) ? 0 : hour12;
    else return (hour12 == 12) ? 12 : hour12 + 12;
}

void display_clock(void) {
    time_t now = time(NULL);
    struct tm *t = localtime(&now);
    const char *days[] = {"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};

    printf("=== Digital Clock ===\n");
    printf("%02d-%02d-%04d (%s)\n",
           t->tm_mday, t->tm_mon + 1, t->tm_year + 1900, days[t->tm_wday]);
    printf("%02d:%02d:%02d\n", t->tm_hour, t->tm_min, t->tm_sec);
    printf("====================\n");
}

void show_task_status(Task tasks[], int count) {
    time_t now = time(NULL);
    struct tm *t = localtime(&now);
    int currentSOD = t->tm_hour * 3600 + t->tm_min * 60 + t->tm_sec;

    for (int i = 0; i < count; i++) {
        int taskSOD = tasks[i].hour24 * 3600 + tasks[i].minute * 60;
        int diff = taskSOD - currentSOD;

        if (diff < 0) diff += 24 * 3600;

        if (diff == 0) {
            if (!tasks[i].reminded) {
```



```c
        if (diff == 0) {
            if (!tasks[i].reminded) {
                printf("🔔 Reminder: %s at %02d:%02d — it's time!\n",
                       tasks[i].name, tasks[i].hour24, tasks[i].minute);
                BEEP();
                tasks[i].reminded = 1;
            } else {
                printf("Task: %-20s | At %02d:%02d | Happening now\n",
                       tasks[i].name, tasks[i].hour24, tasks[i].minute);
            }
        } else {
            int dh = diff / 3600;
            int dm = (diff % 3600) / 60;
            int ds = diff % 60;
            printf("Task: %-20s | At %02d:%02d | Starts in %02d:%02d:%02d\n",
                   tasks[i].name, tasks[i].hour24, tasks[i].minute, dh, dm, ds);
            if (tasks[i].reminded && diff > 60) tasks[i].reminded = 0;
        }
    }
}

int main(void) {
    Task tasks[MAX_TASKS] = {0};
    int taskCount = 0;

    printf("How many tasks do you want to add today (1-%d)? ", MAX_TASKS);
    if (scanf("%d", &taskCount) != 1 || taskCount < 1 || taskCount > MAX_TASKS) {
        fprintf(stderr, "Invalid task count.\n");
        return 1;
    }
    flush_stdin_line();

    for (int i = 0; i < taskCount; i++) {
        printf("Enter task %d name: ", i + 1);
        if (!fgets(tasks[i].name, sizeof(tasks[i].name), stdin)) {
```



```c
            if (scanf("%d %d %2s", &hour12, &minute, ampm) != 3) {
                fprintf(stderr, "Invalid input. Try again.\n");
                flush_stdin_line();
                continue;
            }
            flush_stdin_line();

            char c = tolower((unsigned char)ampm[0]);
            if (hour12 < 1 || hour12 > 12 || minute < 0 || minute > 59 || !(c == 'a' || c == 'p')) {
                printf("Please enter valid time (1–12 hour, 0–59 minute, AM or PM).\n");
                continue;
            }
            tasks[i].hour24 = to_24h(hour12, ampm);
            tasks[i].minute = minute;
            tasks[i].reminded = 0;
            break;
        }
    }

    // Run clock update exactly 6 times
    for (int i = 0; i < 4; i++) {
        CLEAR();
        display_clock();
        printf("----- Today’s Tasks -----\n");
        show_task_status(tasks, taskCount);
        printf("-------------------------\n");

        fflush(stdout);
        PAUSE_1S();
    }

    printf("\nClock stopped after 6 updates.\n");
    return 0;
}
```