DARSHINI P
22011102015

MINI PROJECT REPORT

1. Introduction:

The Digital Clock project aims to create a versatile timekeeping application with multiple functionalities including setting alarms, running timers, and operating a stopwatch. This report outlines the design, implementation, and usage of the Digital Clock application.

2. Objective:

The main objective of the Digital Clock project is to provide users with a comprehensive time management tool that includes features such as alarms, timers, and stopwatches. The project seeks to enhance user productivity and timekeeping accuracy.

3. Features:

- Setting Alarms: Users can set multiple alarms specifying the hour, minute, and second.
- Disabling Alarms: Alarms can be disabled individually if not needed.
- Checking Alarms: The application checks for triggered alarms and notifies the user.
- Printing Alarms: Users can view the list of set alarms along with their status.
- Starting/Stopping Timer: Users can start and stop a timer to track elapsed time.
- Printing Timer: The application displays the current timer status.
- Starting/Stopping Stopwatch: Users can start and stop a stopwatch to measure elapsed time.
- Printing Stopwatch: The application displays the current stopwatch status.

4. Implementation:

The Digital Clock application is implemented in C programming language. It utilizes various data structures such as structs to represent alarms, timers, and stopwatches. Time-related functions from <time.h> library are used for timekeeping operations. The application is menu-driven, allowing users to select desired functionalities.

5. Usage:

- Upon running the application, users are presented with a menu displaying different options.
- Users can select options to set alarms, disable alarms, check alarms, print alarms, start/stop timer, start/stop stopwatch, and exit the application.

- Alarms can be set by specifying the hour, minute, and second. They can be disabled individually if needed.
- The application continuously checks for triggered alarms and notifies the user when an alarm is triggered.
- Timers can be started and stopped to track elapsed time. Stopwatch can also be started and stopped to measure elapsed time.
- Users can print the current status of timers and stopwatches.

6. Code:

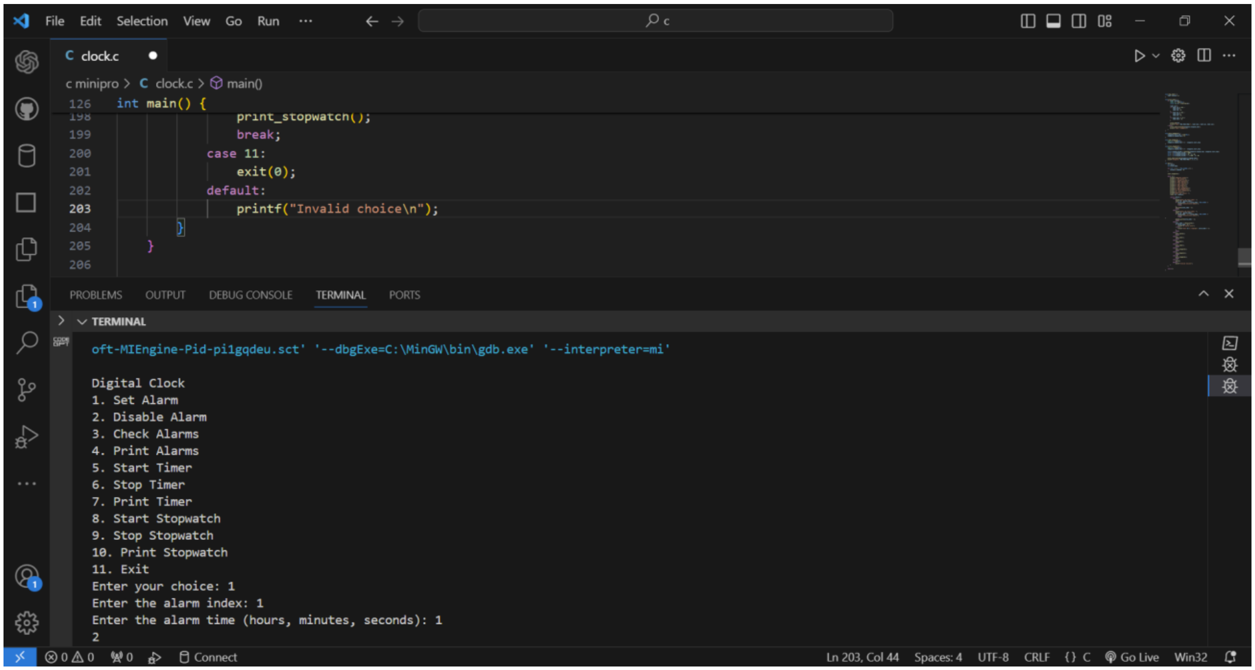
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <conio.h>
#define MAX_ALARMS 5
struct Alarm {
    int hour;
    int min;
    int sec;
    int enabled;
};
struct Timer {
    int hour;
    int min;
    int sec;
    int running;
};
struct Stopwatch {
    time_t start_time;
    time_t elapsed_time;
};
```

```
struct Alarm alarms[MAX_ALARMS];
struct Timer timer;
struct Stopwatch stopwatch;
void print_time(struct tm* tm) {
    printf("%02d:%02d:%02d\n", tm->tm_hour, tm->tm_min, tm->tm_sec);
}
void set_alarm(int index) {
    printf("Enter the alarm time (hours, minutes, seconds): ");
    scanf("%d %d %d", &alarms[index].hour, &alarms[index].min, &alarms[index].sec);
    alarms[index].enabled = 1;
}
void disable_alarm(int index) {
    alarms[index].enabled = 0;
}
int check_alarms() {
    time_t t = time(NULL);
    struct tm* tm = localtime(&t);
  for (int i = 0; i < MAX_ALARMS; i++) {
        if (alarms[i].enabled && alarms[i].hour == tm->tm_hour && alarms[i].min == tm-
>tm_min && alarms[i].sec == tm->tm_sec) {
            return i;
        }
    }
return -1;
}
void print_alarms() {
    printf("Alarms:\n");
    for (int i = 0; i < MAX_ALARMS;i++) {
        printf("%d: %02d:%02d:%02d %s\n", i + 1, alarms[i].hour, alarms[i].min, alarms[i].sec,
alarms[i].enabled ? "Enabled" : "Disabled");
    }
}
void start_timer() {
    timer.running = 1;
    timer.hour = 0;
    timer.min = 0;
    timer.sec = 0;
}
void stop_timer() {
    timer.running = 0;
}
void print_timer() {
    if (timer.running) {
        time_t t = time(NULL);
        struct tm* tm = localtime(&t);

        timer.sec++;
        if (timer.sec >= 60) {
            timer.min++;
            timer.sec = 0;
        }
        if (timer.min >= 60) {
            timer.hour++;
            timer.min = 0;
        }
        if (timer.hour >= 24) {
            timer.hour = 0;
        }
  print_time(tm);
        printf("Timer: %02d:%02d:%02d\n", timer.hour, timer.min, timer.sec);
    } else {
        print_time(localtime(&stopwatch.elapsed_time));
        printf("Timer: Stopped\n");
    }
}
void start_stopwatch() {
    stopwatch.start_time = time(NULL);
    stopwatch.elapsed_time = 0;
}
void stop_stopwatch() {
    time_t t = time(NULL);
    stopwatch.elapsed_time += t - stopwatch.start_time;
}
void print_stopwatch() {
    time_t t = time(NULL);
    stopwatch.elapsed_time += t - stopwatch.start_time;
    double elapsed_seconds = difftime(stopwatch.elapsed_time, stopwatch.start_time);
    int h = (int)elapsed_seconds / 3600;
    int m = (int)elapsed_seconds / 60 - h * 60;
    int s = (int)elapsed_seconds - h * 3600 - m * 60;
 print_time(localtime(&stopwatch.elapsed_time));
    printf("Stopwatch: %02d:%02d:%02d\n", h, m, s);
}
int main() {
    int choice;
```

```
   int alarm_index;
 for (int i = 0; i < MAX_ALARMS; i++) {
       alarms[i].enabled = 0;
   }
start_stopwatch();
while (1) {
       printf("\nDigital Clock\n");
       printf("1. Set Alarm\n");
       printf("2. Disable Alarm\n");
       printf("3. Check Alarms\n");
       printf("4. Print Alarms\n");
       printf("5. Start Timer\n");
       printf("6. Stop Timer\n");
       printf("7. Print Timer\n");
       printf("8. Start Stopwatch\n");
       printf("9. Stop Stopwatch\n");
       printf("10. Print Stopwatch\n");
       printf("11. Exit\n");
       printf("Enter your choice: ");
       scanf("%d", &choice);
    switch (choice) {
           case 1:
               printf("Enter the alarm index: ");
               scanf("%d", &alarm_index);
               if (alarm_index < 1 || alarm_index > MAX_ALARMS) {
                   printf("Invalid alarm index\n");
  break;
               }
```

```
                set_alarm(alarm_index - 1);
                break;
            case 2:
                printf("Enter the alarm index: ");
                scanf("%d", &alarm_index);
                if (alarm_index < 1 || alarm_index > MAX_ALARMS) {
                    printf("Invalid alarm index\n");
                    break;
}
                disable_alarm(alarm_index - 1);
                break;
            case 3:
                alarm_index = check_alarms();
                if (alarm_index < 0) {
                    printf("No alarms set\n");
                } else {
                    printf("Alarm %d is ringing\n", alarm_index + 1);
                }
                break;
            case 4:
                print_alarms();
                break;
            case 5:
                start_timer();
                break;
            case 6:
                stop_timer();
                break;
            case 7:
                print_timer();
                break;
            case 8:
                start_stopwatch();
                break;
            case 9:
                stop_stopwatch();
                break;
            case 10:
                print_stopwatch();
                break;
            case 11:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}
```

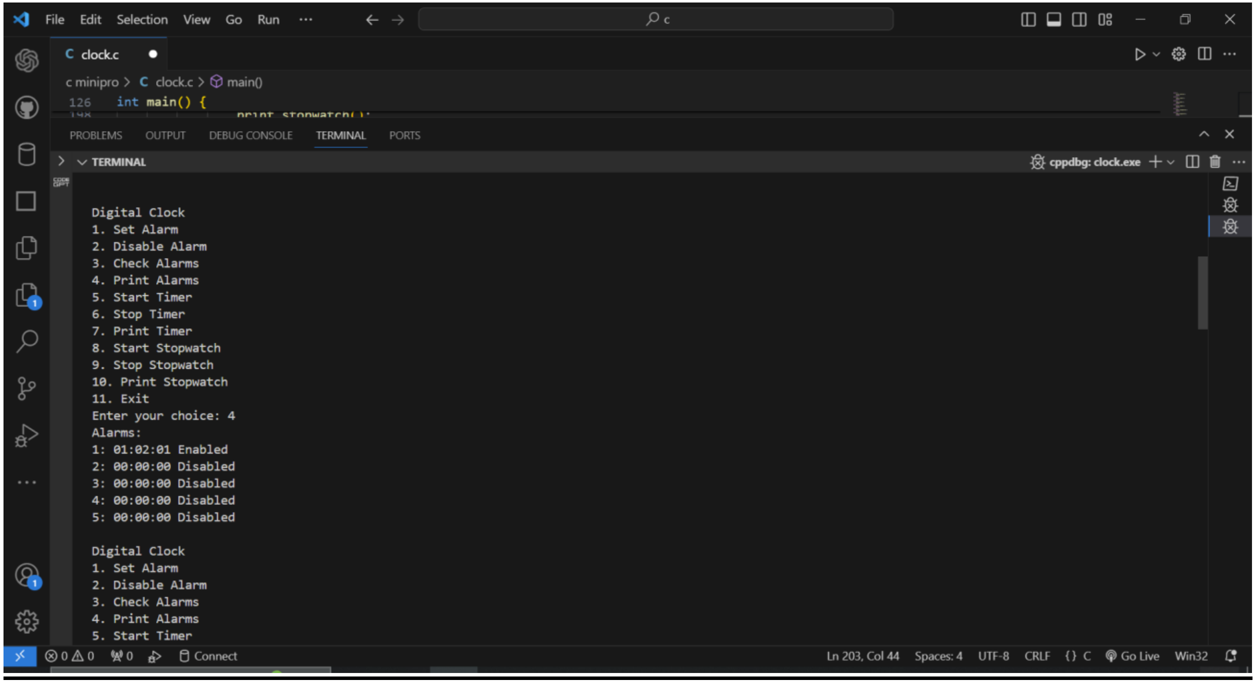
7.Output:



```
int main() {
                print_stopwatch();
                break;
            case 11:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }
```

```
oft-MIEngine-Pid-pi1gqdeu.sct' '--dbgExe=C:\MinGW\bin\gdb.exe' '--interpreter=mi'

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 1
Enter the alarm index: 1
Enter the alarm time (hours, minutes, seconds): 1
2
```

File Edit Selection View Go Run ···
clock.c
c minipro > C clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32
Connect

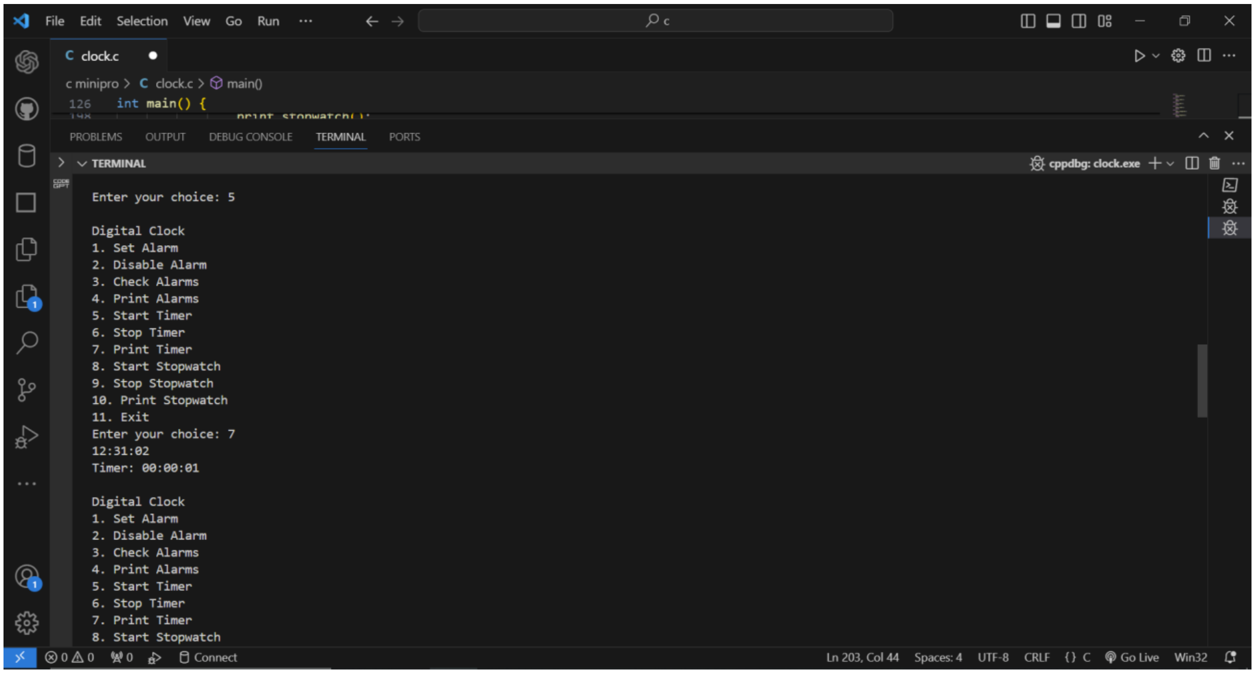


```
int main() {
            print_stopwatch();
```

```
Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 4
Alarms:
1: 01:02:01 Enabled
2: 00:00:00 Disabled
3: 00:00:00 Disabled
4: 00:00:00 Disabled
5: 00:00:00 Disabled

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
```

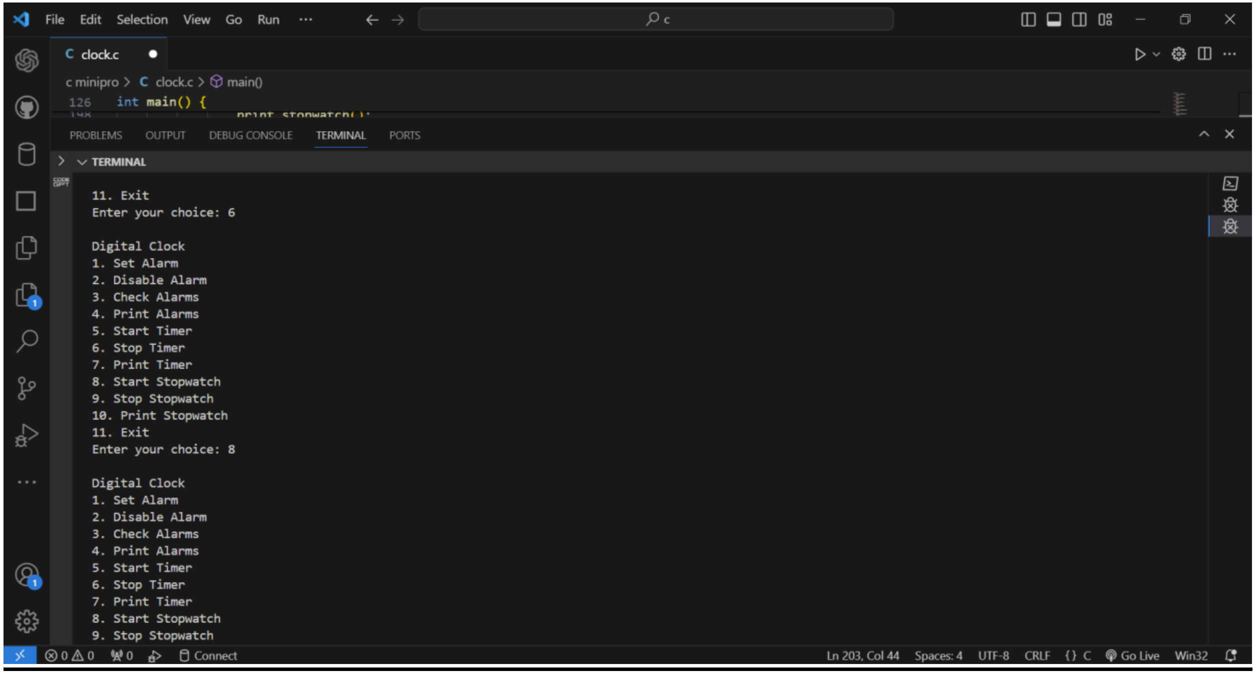
File Edit Selection View Go Run ···
clock.c
c minipro > C clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
cppdbg: clock.exe
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32
Connect

DARSHINI P
22011102015


File Edit Selection View Go Run
clock.c
c minipro > clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
cppdbg: clock.exe
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32
Connect

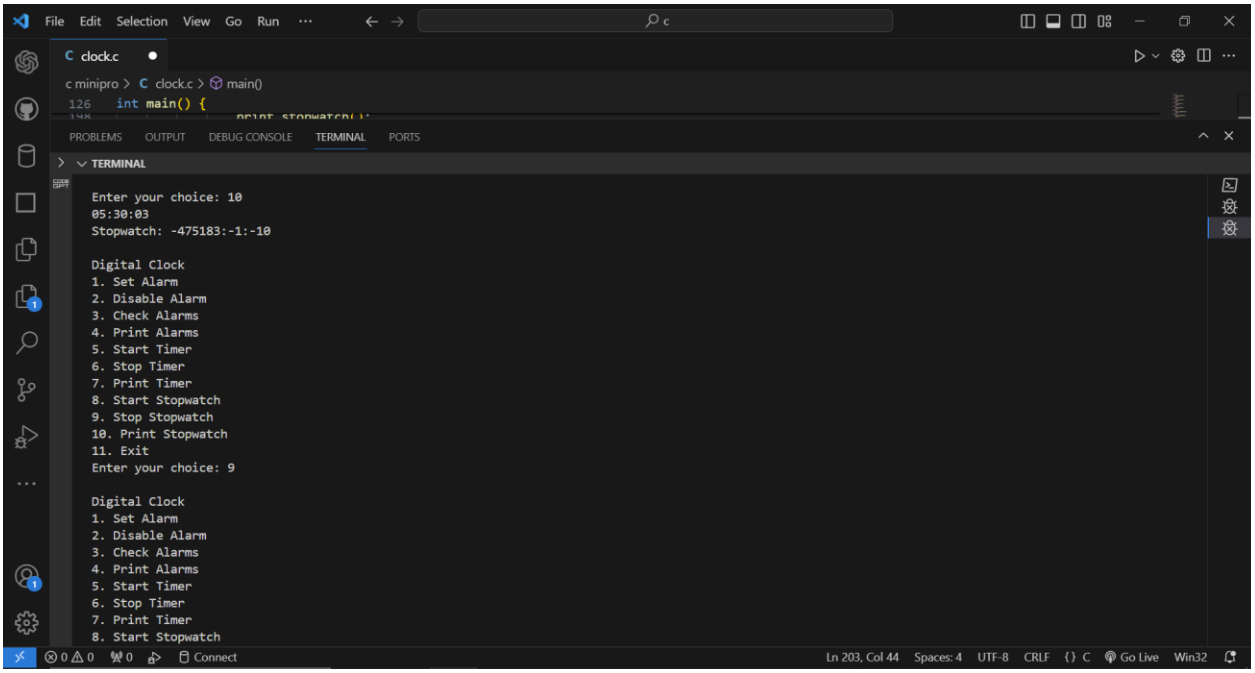
```
int main() {
```

```
Enter your choice: 5

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 7
12:31:02
Timer: 00:00:01

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
```


File Edit Selection View Go Run
clock.c
c minipro > clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32
Connect

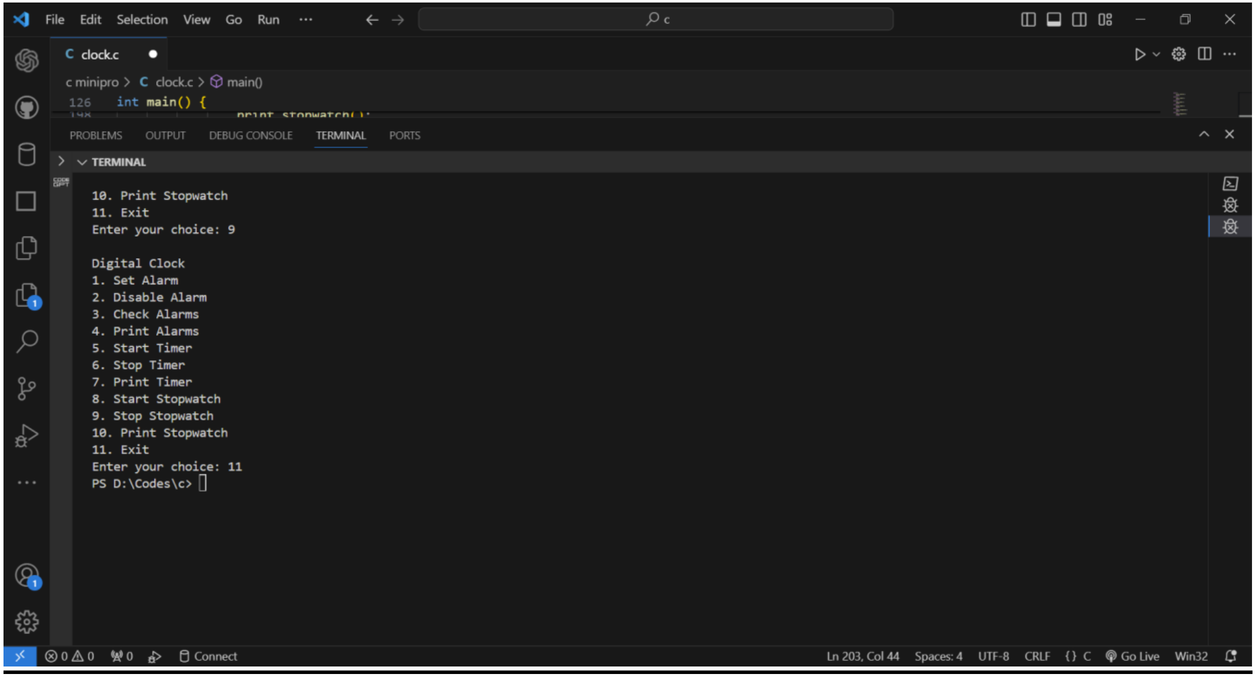
```
int main() {
```

```
11. Exit
Enter your choice: 6

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 8

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
```


File Edit Selection View Go Run ···
c
clock.c
c minipro > C clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
0 0 0 Connect
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32

```
int main() {
```

```
Enter your choice: 10
05:30:03
Stopwatch: -475183:-1:-10

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 9

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
```


File Edit Selection View Go Run ···
c
clock.c
c minipro > C clock.c > main()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TERMINAL
0 0 0 Connect
Ln 203, Col 44 Spaces: 4 UTF-8 CRLF {} C Go Live Win32

```
int main() {
```

```
10. Print Stopwatch
11. Exit
Enter your choice: 9

Digital Clock
1. Set Alarm
2. Disable Alarm
3. Check Alarms
4. Print Alarms
5. Start Timer
6. Stop Timer
7. Print Timer
8. Start Stopwatch
9. Stop Stopwatch
10. Print Stopwatch
11. Exit
Enter your choice: 11
PS D:\Codes\c> 
```

8. Future Enhancements:

- Implementing a graphical user interface (GUI) for improved user interaction.
- Adding functionality to save and load alarm settings.
- Incorporating sound notifications for triggered alarms.
- Enhancing timer and stopwatch functionalities with lap time recording.
- Introducing customization options for alarm sounds and display themes

.

9. Conclusion:

The Digital Clock project provides users with a comprehensive time management solution offering features like alarms, timers, and stopwatches. The application is user-friendly and efficient, aiding users in better timekeeping and productivity enhancement.