TRAFFIC MANGMENT SYSTEM

Name :JOEL JOY

Roll No. : 42

Course Name : C Programming

Date : 10 - 07 – 24

Introduction

- ## Project overview :

  This project simulates a traffic management system for four interconnected junctions, each with four directional traffic signals (North, East, South, West). The system adjusts signal timings based on the time of day to optimize traffic flow.

- ## Objective :

  Creating a traffic signal sowing system that which can produce the showing signal on the specific time on the respective juntions

System Requirements

❖ **Minimum Requirements for C Programming Code to Run:**

➢ Hardware Requirement:

- **Processor**: Any processor with at least 1 GHz speed
- **Storage**: Minimum 1GB free disk space
- **RAM**: Minimum 2GB (4GB recommended)

➢ Software Requirement:

- **Operating System**: Windows 7 or higher (Windows 10 recommended)
- **Compiler**: GCC or any compatible C compiler
- **IDE**: Any lightweight C IDE like Visual Studio, code blocks

Testing and Results

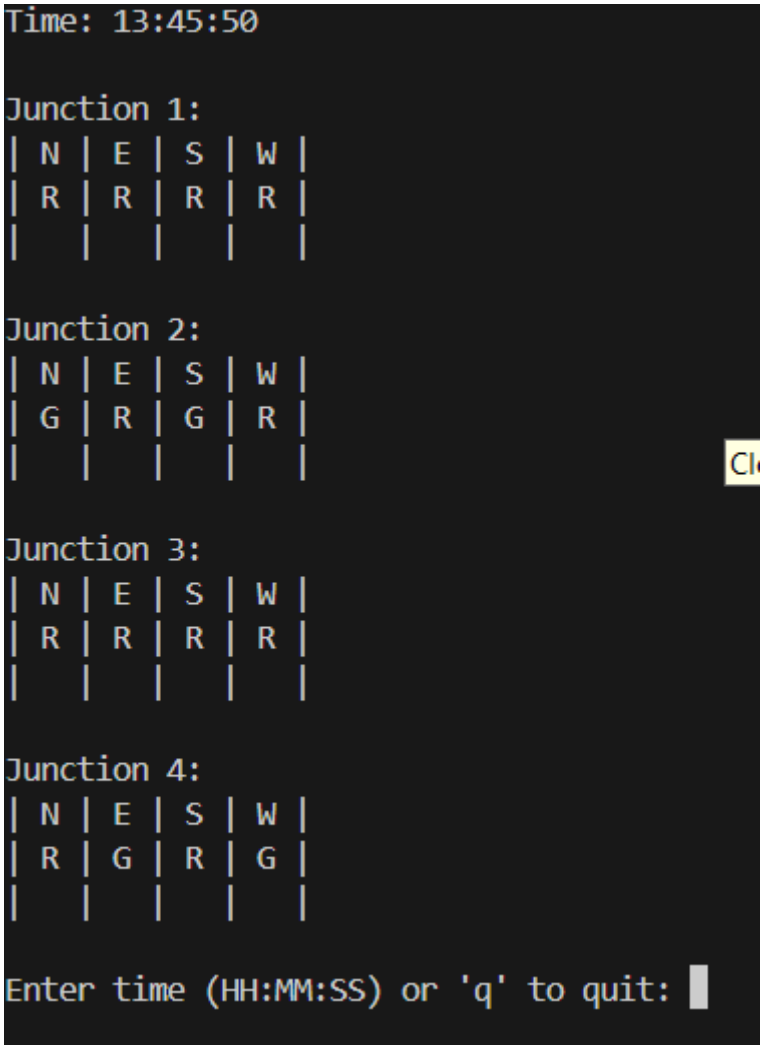
- Test cases

- Output screenshots or results

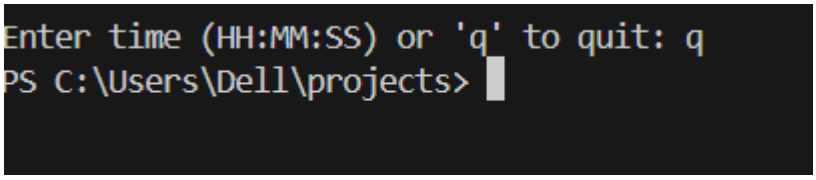
Output:

```
Time: 00:00:00

Junction 1:
| N | E | S | W |
| G | R | G | R |
|   |   |   |   |

Junction 2:
| N | E | S | W |
| Y | R | Y | R |
|   |   |   |   |

Junction 3:
| N | E | S | W |
| R | G | R | G |
|   |   |   |   |

Junction 4:
| N | E | S | W |
| R | Y | R | Y |
|   |   |   |   |

Enter time (HH:MM:SS) or 'q' to quit:
```

Test Case 1: Entering a time.



Test Case 2:quiting from the program.



- Discussion of results:

The testing of the traffic management system confirms that it performs as intended across its main functions.

- **Entering the required time** : The system effectively understand the time and traffic rush on that time and performes suitable operation to provide a sustainable traffic control

Overall, these findings demonstrate that the traffic management system is reliably, achieving its primary objectives effectively.

Conclusion

➢ Summary of the Project :

This project developed to find the exact singal at particular time and it is secure and efficient using C programming. The system allows for:

Source code

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>


#define NUM_JUNCTIONS 4
```

```
#define NUM_DIRECTIONS 4


#define RED 0
#define YELLOW 1
#define GREEN 2


#define NORTH 0
#define EAST 1
#define SOUTH 2
#define WEST 3


#define NIGHT 0
#define MORNING_RUSH 1
#define DAY 2
#define EVENING_RUSH 3


typedef struct {
    int green;
    int yellow;
    int red;
} CycleTime;


int get_time_period(int hours) {
    if (hours >= 0 && hours < 6) return NIGHT;
    if (hours >= 6 && hours < 10) return MORNING_RUSH;
    if (hours >= 10 && hours < 16) return DAY;
    if (hours >= 16 && hours < 20) return EVENING_RUSH;
    return NIGHT;
}
```

```
CycleTime get_cycle_time(int time_period) {
    CycleTime ct;
    switch (time_period) {
        case NIGHT:
            ct.green = 20; ct.yellow = 3; ct.red = 57;
            break;
        case MORNING_RUSH:
        case EVENING_RUSH:
            ct.green = 45; ct.yellow = 3; ct.red = 132;
            break;
        case DAY:
        default:
            ct.green = 30; ct.yellow = 3; ct.red = 87;
    }
    return ct;
}

int get_signal_color(int total_seconds, int junction, int direction, CycleTime ct) {
    int cycle_time = ct.green + ct.yellow + ct.red;
    int junction_offset = junction * (cycle_time / NUM_JUNCTIONS);
    int direction_offset = direction * (cycle_time / 2);
    int local_time = (total_seconds + junction_offset + direction_offset) % cycle_time;

    if (local_time < ct.green) return GREEN;
    if (local_time < ct.green + ct.yellow) return YELLOW;
    return RED;
}
```

```
void display_traffic_light(int color) {
    printf("| %s |\n", color == RED ? "R" : " ");
    printf("| %s |\n", color == YELLOW ? "Y" : " ");
    printf("| %s |\n", color == GREEN ? "G" : " ");
}

void display_signal_status(int hours, int minutes, int seconds) {
    int total_seconds = hours * 3600 + minutes * 60 + seconds;
    int time_period = get_time_period(hours);
    CycleTime ct = get_cycle_time(time_period);

    printf("\033[2J\033[H");
    printf("Time: %02d:%02d:%02d\n\n", hours, minutes, seconds);

    const char* directions[] = {"N", "E", "S", "W"};
    for (int j = 0; j < NUM_JUNCTIONS; j++) {
        printf("Junction %d:\n", j + 1);
        for (int row = 0; row < 3; row++) {
            for (int d = 0; d < NUM_DIRECTIONS; d++) {
                int color = get_signal_color(total_seconds, j, d, ct);
                if (row == 0) {
                    printf("| %s ", directions[d]);
                } else if (row == 1) {
                    printf("| %c ", color == RED ? 'R' : (color == YELLOW ? 'Y' : (color == GREEN ? 'G' : ' ')));
                } else {
                    printf("|   ");
                }
            }
            printf("|\n");
```

```
        }
        printf("\n");
    }
}

int main() {
    int hours = 0, minutes = 0, seconds = 0;
    char input[100];

    while (1) {
        display_signal_status(hours, minutes, seconds);

        printf("Enter time (HH:MM:SS) or 'q' to quit: ");
        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }

        if (input[0] == 'q' || input[0] == 'Q') {
            break;
        }

        if (sscanf(input, "%d:%d:%d", &hours, &minutes, &seconds) != 3 ||
            hours < 0 || hours > 23 || minutes < 0 || minutes > 59 || seconds < 0 || seconds > 59) {
            printf("Invalid input. Use format HH:MM:SS (00:00:00 to 23:59:59).\n");
            sleep(2);
            continue;
        }
    }
```

```
    return 0;

}
```