Understanding The Project Code

Shameem Saqib Niaz (25K-0692)
M. Hamza Siddiqui (25K-0653)

Bus Project Functions - Color Coded (VS Code Style)

clearInputBuffer()

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

Remove any extra character in the code to prevent errors while writing or reading a file. The while loop with an empty { } shows that any left over character till new line or EOF are thrown away.

pauseScreen()

```c
void pauseScreen() {
    printf("\nPress ENTER to continue...");
    clearInputBuffer();
}
```

printHeader()

```c
void printHeader(const char *title) {
    printf("\n==========================================\n");
    printf("   %s\n", title);
    printf("==========================================\n");
}

// ============================================================
//                 LOAD BUS ROUTES FROM FILE
// ============================================================
```

Purpose is to avoid continously printing headers.

loadRoutes()

```
void loadRoutes() {
    FILE *fp = fopen("routes.txt", "r");
    if (!fp) {
        printf("routes.txt missing. No buses loaded.\n");
        busCount = 0;
        return;
    }

    char line[300];
    busCount = 0;

    while (fgets(line, sizeof(line), fp)) {
        if (line[0] == '\n' || line[0] == '\r') continue;

        struct Bus b;
        b.stopCount = 0;

        char *token = strtok(line, ",");
        strcpy(b.id, token);

        token = strtok(NULL, ",");
        b.totalSeats = atoi(token);

        while ((token = strtok(NULL, ",")) != NULL) {
            int stop;
            char t[6];
            sscanf(token, "%d:%5s", &stop, t);
            b.route[b.stopCount] = stop;
            strcpy(b.time[b.stopCount], t);
            b.stopCount++;
        }

        for (int i = 0; i < b.stopCount; i++)
            b.occupancy[i] = 0;

        buses[busCount++] = b;
    }

    fclose(fp);
}

// ============================================================
//                    SAVE ROUTES
// ============================================================
```

ERROR CASE!

cuz reading from file

Makes a sub block of the Struct of bus

Strtok divides the line token by commas into an array kindof

*token points to the first part of the tokenized string i.e bus id such as 6B

strtok actually remembers where is stopped so the null statement just shows that not start to tokenize where we left off.

convert to int

Also stores a time + stop string such as "1:08:00"

Perfectly reads the stop # and time and stores it simultaneously

In an array of routes the stops are stored that shows the route such as stop#1 to 3 to 7 to 5

Stores the times in a times array at the same index as it's stop number

The struct we just formed i.e b is stored into buses[0] and then bus count becomes 1

Closes/Saves the file.

★ The second while loop checks till when it gets a null as return in which case there will be nothing after where the strtok left reading last time

- Save routes func is loaded after the loadBuses function (if admin chooses to). What this does is write the file again.
- When we open a file in write mode it first of all clears the entire file and then writes.

saveRoutes()

```
void saveRoutes() {
    FILE *fp = fopen("routes.txt", "w");
    for (int b = 0; b < busCount; b++) {
        fprintf(fp, "%s,%d", buses[b].id, buses[b].totalSeats);
        for (int i = 0; i < buses[b].stopCount; i++)
            fprintf(fp, ",%d:%s", buses[b].route[i], buses[b].time[i]);
        fprintf(fp, "\n");
    }
    fclose(fp);
}

// ============================================================
//              LOAD STUDENTS FROM FILE
// ============================================================
```

Not Much Logic in this function !

loadStudents()

```
void loadStudents() {
    FILE *fp = fopen("students.txt", "r");
    if (!fp) {
        printf("students.txt not found. Starting with 0 students.\n");
        studentCount = 0;
        return;
    }

    studentCount = 0;
    char line[300];

    while (fgets(line, sizeof(line), fp)) {
        struct Student s;
        char reserved[100], absent[100];

        sscanf(line, "%[^,],%[^,],%[^,],%d,%[^,],%[^\n]",
               s.id, s.name, s.preferredBus, &s.boardingStop,
               reserved, absent);

        int r0, r1, r2, r3, r4, r5, r6;
        int a0, a1, a2, a3, a4, a5, a6;

        sscanf(reserved, "%d %d %d %d %d %d %d",
               &r0, &r1, &r2, &r3, &r4, &r5, &r6);

        sscanf(absent, "%d %d %d %d %d %d %d",
               &a0, &a1, &a2, &a3, &a4, &a5, &a6);

        s.reservedDays[0] = r0; s.absences[0] = a0;
        s.reservedDays[1] = r1; s.absences[1] = a1;
        s.reservedDays[2] = r2; s.absences[2] = a2;
        s.reservedDays[3] = r3; s.absences[3] = a3;
        s.reservedDays[4] = r4; s.absences[4] = a4;
        s.reservedDays[5] = r5; s.absences[5] = a5;
        s.reservedDays[6] = r6; s.absences[6] = a6;

        students[studentCount++] = s;
    }

    fclose(fp);
}

// ============================================================
//              SAVE STUDENTS TO FILE
// ============================================================
```

ERROR CASE !

%[^,]: Scans till the next comma

saveStudents()

```
void saveStudents() {
    FILE *fp = fopen("students.txt", "w");
    for (int i = 0; i < studentCount; i++) {
        struct Student s = students[i];
        fprintf(fp, "%s,%s,%s,%d,%d %d %d %d %d %d %d,%d %d %d %d %d %d %d\n",
                s.id, s.name, s.preferredBus, s.boardingStop,
                s.reservedDays[0], s.reservedDays[1], s.reservedDays[2],
                s.reservedDays[3], s.reservedDays[4], s.reservedDays[5], s.reservedDays[6],
                s.absences[0], s.absences[1], s.absences[2],
                s.absences[3], s.absences[4], s.absences[5], s.absences[6]);
    }
    fclose(fp);
}

// ============================================================
//            ID Exists?
// ============================================================
```

idExists()

```
int idExists(char id[]) {
    for (int i = 0; i < studentCount; i++)
        if (strcmp(students[i].id, id) == 0) return 1;
    return 0;
}

// ============================================================
//              FIND BUS INDEX
// ============================================================
```

findBusIndex()

```
int findBusIndex(char busID[]) {
    for (int i = 0; i < busCount; i++)
        if (strcmp(buses[i].id, busID) == 0) return i;
    return -1;
}
```

Return bus index else -1 which is used to make error cases

```
// ============================================================
//            STOP EXISTS?
// ============================================================
```

stopExistsInBus()

```
int stopExistsInBus(int busIndex, int stopID) {
    for (int i = 0; i < buses[busIndex].stopCount; i++)
        if (buses[busIndex].route[i] == stopID) return 1;
    return 0;
}
```

returns 1 if The bus has stop ID in its route
else returns 0

```
// ============================================================
//          REGISTER STUDENT
// ============================================================
```

registerStudent()

```
void registerStudent() {
    printHeader("Register Student");

    struct Student s;

    printf("Enter Student ID (e.g., 25K-0653): ");
    scanf("%19s", s.id);

    if (idExists(s.id)) {
        printf("ID already exists!\n");
        pauseScreen();
        return;
    }

    printf("Enter student name: ");
    clearInputBuffer();
    fgets(s.name, 50, stdin);
    s.name[strcspn(s.name, "\n")] = 0;

    printf("Enter preferred bus ID: ");
    scanf("%9s", s.preferredBus);

    int b = findBusIndex(s.preferredBus);
    if (b == -1) {
        printf("Bus not found.\n");
        pauseScreen();
        return;
    }

    printf("Stops:\n");
    for (int i = 0; i < buses[b].stopCount; i++)
        printf("%d at %s\n", buses[b].route[i], buses[b].time[i]);

    printf("Enter boarding stop: ");
    scanf("%d", &s.boardingStop);

    if (!stopExistsInBus(b, s.boardingStop)) {
        printf("Stop not valid.\n");
        pauseScreen();
        return;
    }

    printf("Enter weekly schedule (Sun..Sat): ");
    for (int i = 0; i < 7; i++) scanf("%d", &s.reservedDays[i]);

    for (int i = 0; i < 7; i++) s.absences[i] = 0;

    students[studentCount++] = s;
    saveStudents();

    printf("Student registered.\n");
    pauseScreen();
}

// ============================================================
//          MARK ABSENT
// ============================================================
```

* Id Exists function is called and helps check whether id is unique or not

strcspn here removes the \n that fgets takes

ERROR CASE !

Used for a Better UI experience for the student (prints stops name with time)

ERROR CASE

StudentCount will increment after saving students[0]

markAbsent()

```
void markAbsent() {
    printHeader("Mark Absent");

    char id[20];
    int day;

    printf("Enter student ID: ");
    scanf("%19s", id);

    printf("Day (0-6): ");
    scanf("%d", &day);

    for (int i = 0; i < studentCount; i++) {
        if (strcmp(students[i].id, id) == 0) {
            students[i].absences[day] = 1;
            saveStudents();
            printf("Marked absent.\n");
            pauseScreen();
            return;
        }
    }

    printf("Student not found.\n");
    pauseScreen();
}

// ============================================================
//    CALCULATE OCCUPANCY FOR GIVEN DAY
// ============================================================
```

Inputs ID

Inputs day

Iterates through each student till an ID is found

When found absent of the day for the student is found.

(Updates students.txt)

calculateDailyOccupancy()

```
void calculateDailyOccupancy(int day) {
    for (int b = 0; b < busCount; b++)
        for (int i = 0; i < buses[b].stopCount; i++)
            buses[b].occupancy[i] = 0;

    for (int s = 0; s < studentCount; s++) {
        if (students[s].reservedDays[day] == 0) continue;
        if (students[s].absences[day] == 1) continue;

        int b = findBusIndex(students[s].preferredBus);
        if (b == -1) continue;

        int add = 0;
        for (int i = 0; i < buses[b].stopCount; i++) {
            if (buses[b].route[i] == students[s].boardingStop) add = 1;
            if (add) buses[b].occupancy[i]++;
        }
    }
}

// ============================================================
//          VIEW REPORT
// ============================================================
```

Turns all occupancy to zero

These condition check if a student is absent so the code following this will not execute incase a student is absent.

Finds the bus index for further use

Iterates through till it checks all stops for a student boarding the stop and increments by one

viewReport()

```
void viewReport() {
    printHeader("Bus Occupancy Report");

    int day;
    printf("Day (0=Sun..6=Sat): ");
    scanf("%d", &day);

    calculateDailyOccupancy(day);

    for (int b = 0; b < busCount; b++) {
        printf("\nBus %s:\n", buses[b].id);
        for (int i = 0; i < buses[b].stopCount; i++)
            printf("Stop %d at %s ? %d/%d\n",
                   buses[b].route[i], buses[b].time[i],
                   buses[b].occupancy[i], buses[b].totalSeats);
    }

    pauseScreen();
}

// ============================================================
//          CHECK AVAILABILITY
// ============================================================
```

will set occupancy to correct amount for further usage

checkAvailability()

```
void checkAvailability() {
    printHeader("Check Availability");

    char busID[10];
    int stop, day;

    printf("Bus ID: ");
    scanf("%9s", busID);

    printf("Stop: ");
    scanf("%d", &stop);

    printf("Day: ");
    scanf("%d", &day);

    calculateDailyOccupancy(day);

    int b = findBusIndex(busID);
    if (b == -1) {
        printf("Bus not found.\n");
        pauseScreen();
        return;
    }

    for (int i = 0; i < buses[b].stopCount; i++) {
        if (buses[b].route[i] == stop) {
            printf("%d/%d seats used.\n",
                   buses[b].occupancy[i], buses[b].totalSeats);
            pauseScreen();
            return;
        }
    }
}

// ============================================================
//          SUGGEST ALTERNATIVE BUS
// ============================================================
```

suggestAlternative()

```
void suggestAlternative() {
    printHeader("Alternative Bus");

    char id[20];
    int day;

    printf("Enter student ID: ");
    scanf("%19s", id);

    printf("Enter day: ");
    scanf("%d", &day);

    calculateDailyOccupancy(day);

    int sIndex = -1;
    for (int i = 0; i < studentCount; i++)
        if (strcmp(students[i].id, id) == 0) sIndex = i;

    if (sIndex == -1) {
        printf("Not found.\n");
        pauseScreen();
        return;
    }
```

ERROR CASE !

```
    struct Student s = students[sIndex];
    int b = findBusIndex(s.preferredBus);

    int pos = -1;
    for (int i = 0; i < buses[b].stopCount; i++)
        if (buses[b].route[i] == s.boardingStop) pos = i;

    if (buses[b].occupancy[pos] < buses[b].totalSeats) {
        printf("Preferred bus has seat.\n");
        pauseScreen();
        return;
    }
```

→ Student's data is fetched

→ Fetches Bus index

→ Fetches an integer value of occupancy at a certain stop which is fetched by pos before this.

★ Logically this is an else statement.

```
    printf("Preferred bus full. Searching...\n");

    for (int i = 0; i < busCount; i++) {
        if (i == b) continue;
```

(Bus already full so no need to check)

```
        for (int j = 0; j < buses[i].stopCount; j++) {
            if (buses[i].route[j] == s.boardingStop &&
                buses[i].occupancy[j] < buses[i].totalSeats) {

                printf("Take Bus %s at %s\n",
                       buses[i].id, buses[i].time[j]);
                pauseScreen();
                return;
            }
        }
    }

    printf("No alternative.\n");
    pauseScreen();
}

// ============================================================
//                    ADMIN PANEL
// ============================================================
```

adminPanel()

```
void adminPanel() {
    char pass[20];
    printHeader("Admin Login");

    printf("Password: ");
    scanf("%19s", pass);

    if (strcmp(pass, ADMIN_PASSWORD) != 0) {
        printf("Wrong password.\n");
        pauseScreen();
        return;
    }

    int c;
    while (1) {
        printHeader("Admin Menu");
        printf("1. View Buses\n");
        printf("2. Save Routes\n");
        printf("3. Back\n");
        printf("Enter: ");
        scanf("%d", &c);

        if (c == 1) {
            for (int i = 0; i < busCount; i++)
                printf("%s (%d seats, %d stops)\n",
                       buses[i].id, buses[i].totalSeats, buses[i].stopCount);
            pauseScreen();
        }
        else if (c == 2) {
            saveRoutes();
            pauseScreen();
        }
        else if (c == 3) break;
    }
}

// ============================================================
//                          MAIN
// ============================================================
```

main()

```c
int main() {
    loadRoutes();
    loadStudents();

    while (1) {
        printHeader("University Bus Reservation System");
        printf("1. Register Student\n");
        printf("2. Mark Absent\n");
        printf("3. Check Availability\n");
        printf("4. Suggest Alternative\n");
        printf("5. View Report\n");
        printf("6. Admin Panel\n");
        printf("7. Exit\n");
        printf("Enter choice: ");

        int c;
        scanf("%d", &c);

        if      (c == 1) registerStudent();
        else if (c == 2) markAbsent();
        else if (c == 3) checkAvailability();
        else if (c == 4) suggestAlternative();
        else if (c == 5) viewReport();
        else if (c == 6) adminPanel();
        else if (c == 7) {
            saveStudents();
            printHeader("Goodbye!");
            return 0;
        }
        else printf("Invalid choice.\n");
    }
}
```