CODIAC: SELF-LEARNING AND QUIZ ENVIRONMENT

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

SRILEKHYA TURLAPATI - 1602-19-737-181

TEJASREE ANNAPURNAREDDY – 1602-19-737-179

B. RANEMMA – 1602-19-737-153

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad - 500 031

2020

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

2020

Hyderabad - 500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **SRILEKHYA TURLAPATI**, **TEJASREE REDDY ANNAPUREDDY and B. RANEMMA** bearing hall ticket numbers, 1602-19-737-181, 1602-19-737-179 and 1602-19-737-153 respectively, hereby declare that the project report entitled **CODIAC: SELF-LEARNING AND QUIZ ENVIRONMENT** is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

SRILEKHYA TURLAPATI
1602-19-737-181

TEJASREE REDDY ANNAPUREDDY
1602-19-737-179

B. RANEMMA
1602-19-737-153

(Faculty In-Charge) (Head, Dept. of IT)

ACKNOWLEDGEMENTS

Our Mini Project would not have been successful without the help of several people. We are extremely thankful to our college, **Vasavi College of Engineering, Hyderabad** for providing the opportunity to implement our project, **"CODIAC: A Self-Learning and Quiz Environment"**.

We would like to express our gratitude to **Ms. Divya Lingineni**, Assistant Professor, Department of Information Technology, Vasavi College of Engineering and **Dr. Ramesh Vasappanavara**, Professor, Department of Information Technology, Vasavi College of Engineering, for their esteemed guidance, moral support and invaluable advice provided by them for the success of the Mini Project.

Sincerely,

SRILEKHYA TURLAPATI 1602-19-737-181

TEJASREE REDDY ANNAPUREDDY 1602-19-737-179

B. RANEMMA 1602-19-737-153

ABSTRACT

The concept of quizzes is currently very popular among education circles. Quizzes contribute to the growth and knowledge of an individual. The intention behind the consideration of this project is to design and implement a platform for individuals to learn different programming languages/concepts, namely, C, Python and OOPs Concepts and to be able to test his/her knowledge through an online quiz. This quiz consists of multiple-choice questions with points that will be added after each correct answer. Other than this, we have also included an optional feature for users to assess themselves after each topic learnt. We are providing this opportunity to allow users to test the knowledge they have gained and offer the chance to re-learn the topic studied. This will create a sense of enthusiasm of wanting to learn and having fun at the same time.

TABLE OF CONTENTS

1. Introduction 1

1.1. About The Project 1

1.2. Project Domain 1

1.2.1. Technical Domain 1

1.2.2. Functional Domain 1

1.3. Features 2

1.3.1. Learn 2

1.3.2. Mini-Assessment 2

1.3.3. Quiz 2

1.3.4. Leader Board 3

2. Technology 4

2.1. Software Requirements 4

2.2. Hardware Requirements 4

3. Proposed Work 5

3.1. Design 5

3.1.1. User Use Cases 5

3.1.1.1. Register 5

3.1.1.2. Login 6

3.1.1.3. Learn A Concept 6

3.1.1.4. Take A Quiz 6

3.1.1.5. View Points Leader Board 6

3.1.2. Admin Use Cases 7

3.1.2.1. Login 7

3.1.2.2. View User Table 7

3.1.2.3. Edit Users Details 8

3.1.2.4. Delete User 8

3.1.2.5. View Points Leader Board 8

3.2. Implementation 8

3.2.1. Module-Wise Code 9
3.2.1.1. Register And Login 9
3.2.1.2. Learn 19
3.2.1.3. Mini-Assessment 25
3.2.1.4. Main Quiz 27
3.2.1.5. Banner 30
3.2.2. Github/Folder Structure 34
3.3. Testing 35
3.3.1. Test Plan 35
3.3.2. User Test Cases 35
3.3.2.1. Register 35
3.3.2.2. Login 36
3.3.2.3. Learn A Concept 37
3.3.2.4. Take A Quiz 39
3.3.2.5. View Points Leaderboard 39
3.3.3. Admin Test Cases 40
3.3.3.1. Login 40
3.3.3.2. View User Table 41
3.3.3.3. Edit Users Details 41
3.3.3.4. Delete User 42
3.3.3.5. View Points Leader Board 42
4. Results 44
4.1. User Test Case Results 44
4.2. Admin Test Case Results 52
5. Additional Knowledge Acquired 57
6. Conclusion And Future Work 58
7. References 59

1. INTRODUCTION

1.1. ABOUT THE PROJECT

"CODIAC – A Self-Learning and Quiz Environment" is a console-based C Project which acts as a learning platform for users of all age groups to improve their skill set. The intent behind this project is to inculcate the zeal to learn important Computer Science based concepts, such as C, Python and OOPs.

1.2. PROJECT DOMAIN

The domain of the project is the targeted subject area of a computer program. It is a term most commonly used in software engineering. Formally, it represents the target subject of a specific programming project, whether narrowly or broadly defined. To be concise, a **domain** in the realm of software engineering commonly refers to the subject area on which the application is intended to apply. Within a domain, there are two categories:

1. Technical Domain
2. Functional Domain

1.2.1. TECHNICAL DOMAIN

This project falls under the domain of a Console Application. A console application is a program designed to be used via a text-only computer interface, such as a text terminal, the command line interface of some operating systems or the text-based interface included with most GUI (Graphical User Interface) operating systems.

1.2.2. FUNCTIONAL DOMAIN

'CODIAC' comes under the Learning and Development domain, because we are aiming to teach and help users improve their skills. As we are providing knowledge to the user, we fall under a subdomain called Cognitive.

1.3. FEATURES

Our vision for this project was to provide a platform for users to practise Self-Learning in a controlled environment. We mainly focussed on the Learning and the Quiz module, because that was the base of our project.

Another component we wanted to keep emphasis on was the points leader board. This is because only learning won't satisfy all users. Hence, we thought of keeping a ranking system to encourage the competitive streak in all users.

To be simple, the features that we have offered are a chance for users to learn, assess themselves, take a quiz and view the points leader board.

1.3.1. LEARN

The Learning section of CODIAC was created on the basis of instilling independence in the user's learning style. The concepts that we have focussed on are C Language, Python Language and Object-Oriented Programming System as they are significant subjects which are very applicable in today's life. We aim to provide the user with information such as a brief introduction on the topic, syntax and short examples to give an idea about the topic.

1.3.2. MINI-ASSESSMENT

We have provided a Mini-Assessment portion within the Learning module to give the user an option to test him/herself based on the topic read. This won't count toward the ranking system, but it will allow the user a chance to test their knowledge on what they have read. If a user scores less, then he/she can go back and re-read the topic and re-take the self-assessment. The Mini-Assessment feature was provided for this reason, that is, to allow the user to rectify his/her mistakes.

1.3.3. QUIZ

The Quiz module is one of the most important parts of the entire project. Our entire program is modelled around this idea. As said before, we wanted to improve

not just our skills but also others as well. Hence, we planned to set up a question bank and extract 20 questions from them. Each question will be appointed 1 point and rewarded to the user. After each attempt of the quiz, we will display the points scored and this will get added to the total score.

1.3.4. LEADER BOARD

The ranking system was a decision taken to draw out the ambition in all users. We want to encourage everyone to learn and this feature would help the objective.

2. TECHNOLOGY

All computer software needs certain hardware components or other software resources to be present, in order for computers to be used efficiently. These prerequisites are known as System Requirements. Within this, we have two types – Software Requirements and Hardware Requirements.

2.1. SOFTWARE REQUIREMENTS

Software Requirements deal with defining the software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These preconditions are generally not included in the software installation package and need to be installed separately.

In order to use CODIAC, one should have the following:

- **Operating System:** Windows 7 and above
- **C Compiler:** GNU Compiler Collection (GCC)
- **Editor:** Any text editor (preferably Visual Studio Code)

2.2. HARDWARE REQUIREMENTS

Hardware requirements refer to the common set requirements defined by any operating system or software application and are usually the physical computer resources. In this, we look into the architecture, processing power, memory, secondary memory, display adapter and peripherals.

In order to use CODIAC, one should have the following:

- **Processor:** Intel Core i5 and above
- **Memory:** 8 GB RAM

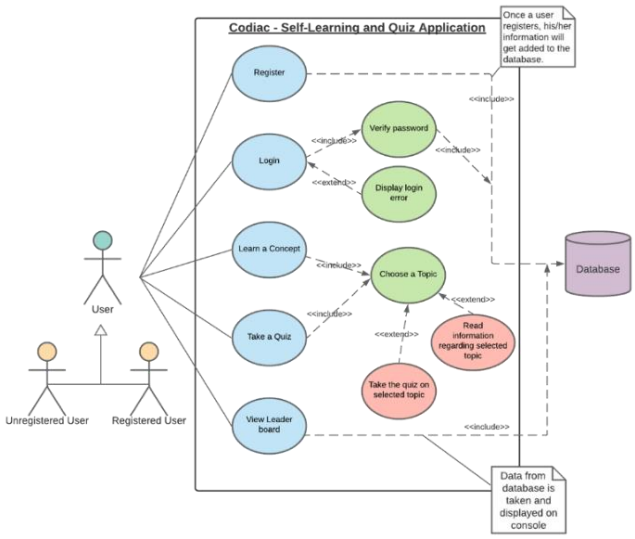
3. PROPOSED WORK

3.1. DESIGN

Our approach in designing CODIAC was to divide our users into two groups – the users and the Admin. The user is a day-to-day consumer whose only goal is to learn in a safe environment. The Admin's end goal is to maintain CODIAC's database and cater to the demand of any requests of the user. The user can approach the Admin for changes in their credentials, except for the username.

3.1.1. USER USE CASES

The user is a person who has limited access to CODIAC. He/she will have 5 functionalities: Register, Login, Learn, Mini-Assessment and the Main Quiz.



3.1.1.1. REGISTER

Any user can register as a user in CODIAC. First, the system prompts the user to enter his/her details. Once, the user enters the required details (first name, last name, username and password), the system checks whether the username checks or not and whether the password meets the prerequisites. If the username exists, the system displays a message to choose another username. If the password is not valid, it will display the respective message of what to fix.

3.1.1.2. LOGIN

In order for a user to login to CODIAC, he/she must be a registered user. First, the system will prompt the user to enter his/her credentials (username and password). Once, the user enters his/her credentials, we check whether it matches with the information in the database. If a match is found, the user gets access to CODIAC – A Self-Learning and Quiz Environment.

3.1.1.3. LEARN A CONCEPT

Once the user is logged in, he/she can choose to learn a concept, from either C, Python or OOPs. After choosing the concept, the system displays a list of all the topics relevant to the concept. Then, after choosing the topic, the user will get to see information regarding the topic. This information includes definitions, syntax and examples. After learning a topic, the user can take a mini-assessment for testing their knowledge. This is optional and won't count for the leader board points system.

3.1.1.4. TAKE A QUIZ

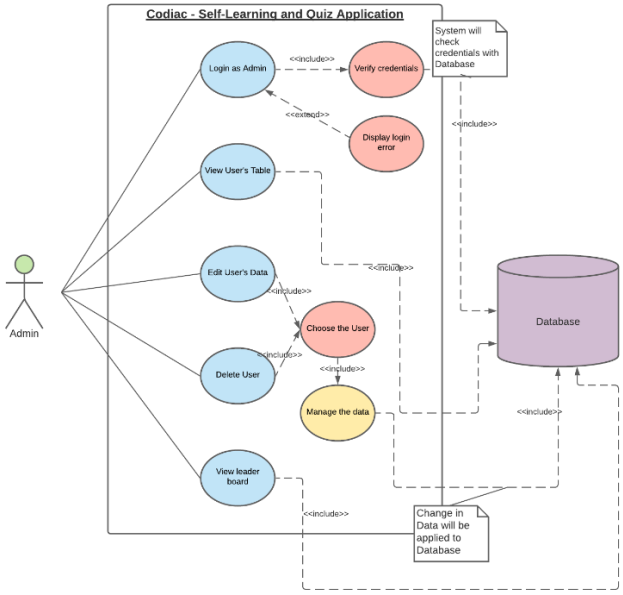
After logging in, the user can choose to take the main quiz. First, the user should choose the concept they want to be tested in and then they can write the quiz. The points scored in that particular taking of the quiz will be displayed for the user at the end. These points will then be added to the total score and displayed on the leader board.

3.1.1.5. VIEW POINTS LEADER BOARD

Once the user logs in, he/she can view the points leader board. The leader board is a rank-wise table containing all the users who have attempted the quiz and their total score. We have implemented this feature for creating a sense of enthusiasm of wanting to learn.

3.1.2. ADMIN USE CASES

The Admin is a user who has access to all information, including the user's credentials and even the leader board. The Admin also has the license to edit any user's details and even delete a user if needed.



3.1.2.1. LOGIN

The Admin can login if he/she has the required credentials. The system first verifies whether the credentials entered match the Admin credentials. If they do, then the Admin gets complete access to CODIAC. If they don't match, the required message is displayed on the console.

3.1.2.2. VIEW USER TABLE

After the Admin logs in, he/she can view all the information of all registered users. This information includes first name, last name, username, password and total amount of points scored so far. We have included this functionality for ensuring easy access of user information for the Admin.

3.1.2.3. EDIT USERS DETAILS

Once logged in as the Admin, he/she can choose to edit any users' details. First, we load the user information from the database and the Admin can choose the user whose details they want to edit. Then, the system displays the chosen user's details. From this, the Admin should choose the data they want to edit. After editing, the old data in the database is replaced with the new data. A point to be noted is that the Admin cannot edit any user's username, because of future damage in rankings. Hence, that option is disabled.

3.1.2.4. DELETE USER

Only the Admin has the authority to delete a user. The Admin may have different reasons for deleting a user. It can be because of malpractice or inactivity. Whatever the reason, if the Admin needs to delete a user, he/she has to enter the username of the user they want to remove from CODIAC permanently. Then, the system deletes the chosen user data from the database.

3.1.2.5. VIEW POINTS LEADER BOARD

The Admin also has the power to view the points leader board. The Admin does not have the authorization to change any ranking in the points leader board. It is a read-only feature for the Admin.

3.2. IMPLEMENTATION

Based on the use cases, we have implemented this project by dividing the work into 5 modules – Register and Login, Learn, Mini-Assessment, Main Quiz and Banner.

3.2.1. MODULE-WISE CODE

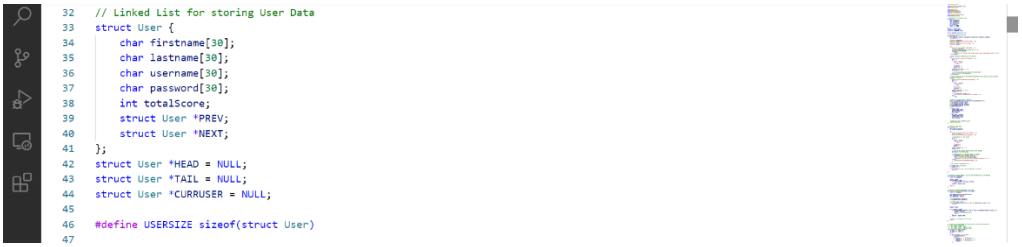
3.2.1.1. REGISTER AND LOGIN

The Register and Login Module contains all the functions pertaining to User Registration and Login and also contains all the functions related to the Admin and its use cases.

Function definitions and importing Learn and MainQuiz module

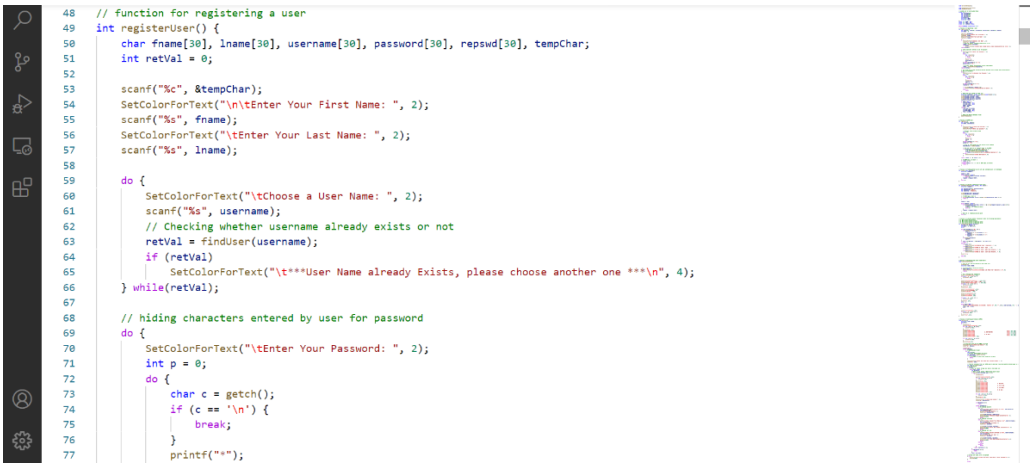
```
// Register and Login Module

// Including Learn and MainQuiz Modules
#include "Learn.h"
#include "MainQuiz.h"

// function declarations
int registerUser();
int loginUser();

int findUser(char *);
int checkUserCredentials(char *, char *);
int validatePassword(char *);

void displayLeaderBoard();

void editData();
void deleteUser();
int showUsersList();

void updateUserScore(char[], int);
void sortAllUsers();

void loadUsers();
void deleteAllNodes();
void addUserToDatabase();
void rewriteToDatabase();

void showUserSelectionScreen();
void showAdminScreen();
```

Structure for making a Linked List for storing all user data

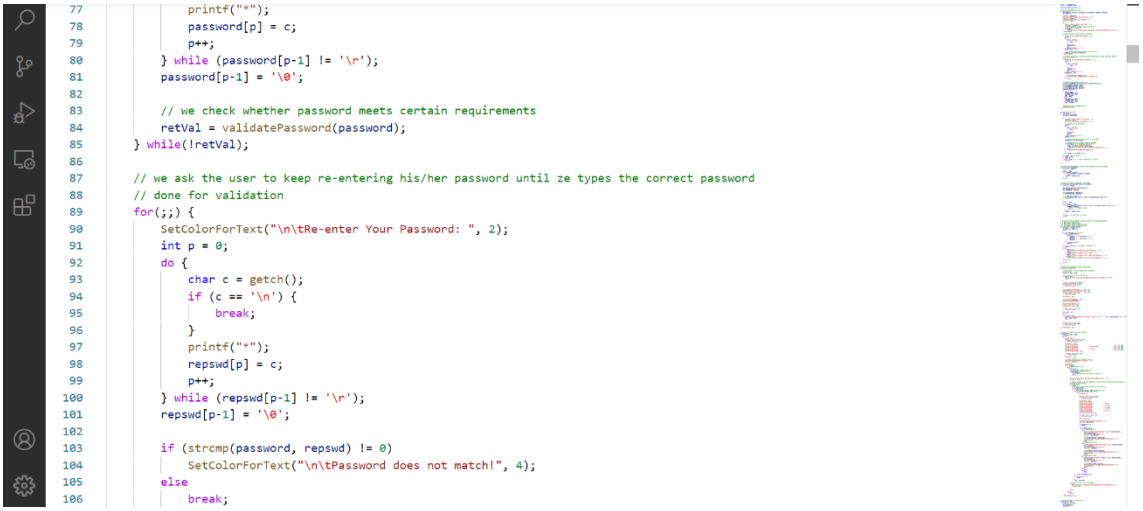


```
// Linked List for storing User Data
struct User {
    char firstname[30];
    char lastname[30];
    char username[30];
    char password[30];
    int totalScore;
    struct User *PREV;
    struct User *NEXT;
};
struct User *HEAD = NULL;
struct User *TAIL = NULL;
struct User *CURRUSER = NULL;

#define USERSIZE sizeof(struct User)

```

Function: void registerUser()



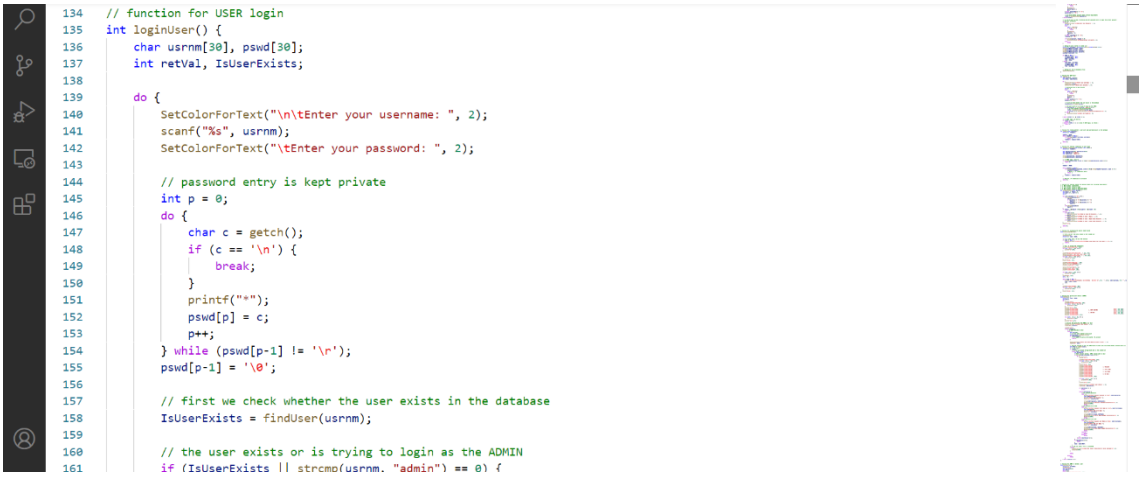
```
// function for registering a user
int registerUser() {
    char fname[30], lname[30], username[30], password[30], repswd[30], tempChar;
    int retVal = 0;

    scanf("%c", &tempChar);
    SetColorForText("\n\tEnter Your First Name: ", 2);
    scanf("%s", fname);
    SetColorForText("\tEnter Your Last Name: ", 2);
    scanf("%s", lname);

    do {
        SetColorForText("\tChoose a User Name: ", 2);
        scanf("%s", username);
        // Checking whether username already exists or not
        retVal = findUser(username);
        if (retVal)
            SetColorForText("\t***User Name already Exists, please choose another one ***\n", 4);
    } while(retVal);

    // hiding characters entered by user for password
    do {
        SetColorForText("\tEnter Your Password: ", 2);
        int p = 0;
        do {
            char c = getch();
            if (c == '\n') {
                break;
            }
            printf("*");
```



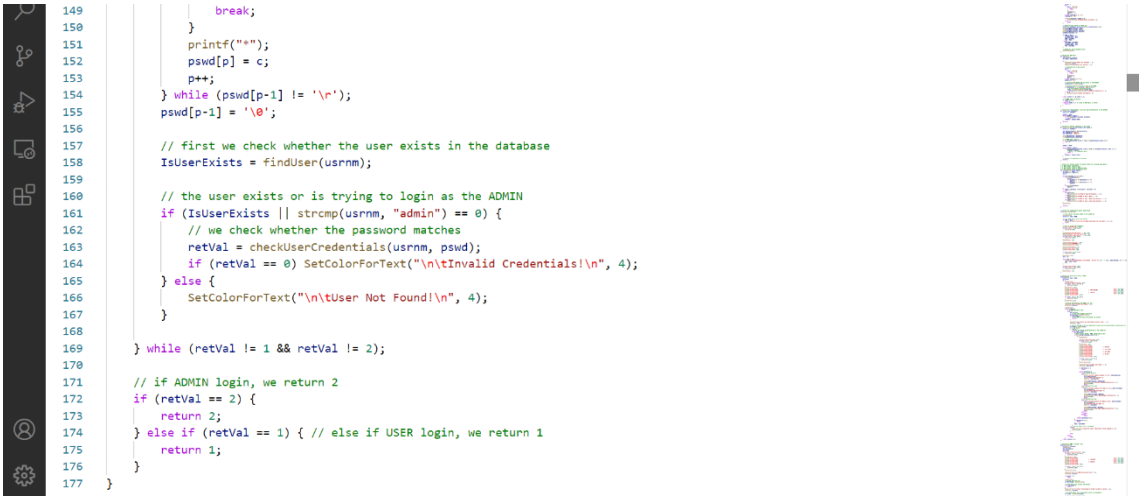
```
            printf("*");
            password[p] = c;
            p++;
        } while (password[p-1] != '\r');
        password[p-1] = '\0';

        // we check whether password meets certain requirements
        retVal = validatePassword(password);
    } while(!retVal);

    // we ask the user to keep re-entering his/her password until ze types the correct password
    // done for validation
    for(;;) {
        SetColorForText("\n\tRe-enter Your Password: ", 2);
        int p = 0;
        do {
            char c = getch();
            if (c == '\n') {
                break;
            }
            printf("*");
            repswd[p] = c;
            p++;
        } while (repswd[p-1] != '\r');
        repswd[p-1] = '\0';

        if (strcmp(password, repswd) != 0)
            SetColorForText("\n\tPassword does not match!", 4);
        else
            break;
```

```
        if (strcmp(password, repswd) != 0)
            SetColorForText("\n\tPassword does not match!", 4);
        else
            break;
    }

    // adding new user created to linked list
    struct User *userNode = (struct User*)malloc(sizeof(struct User));
    strcpy(userNode->firstname, fname);
    strcpy(userNode->lastname, lname);
    strcpy(userNode->username, username);
    strcpy(userNode->password, password);
    userNode->totalScore = 0;

    if (HEAD == NULL) {
        userNode->PREV = NULL;
        userNode->NEXT = NULL;
        HEAD = userNode;
        TAIL = HEAD;
    } else {
        TAIL->NEXT = userNode;
        userNode->PREV = TAIL;
        userNode->NEXT = NULL;
        TAIL = userNode;
    }

    // adding user data to database (file)
    addUserToDatabase();
}
```

Function: void loginUser():

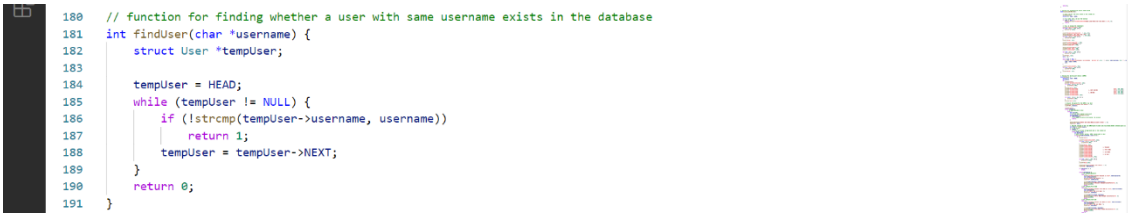


```
// function for USER login
int loginUser() {
    char usrnm[30], pswd[30];
    int retVal, IsUserExists;

    do {
        SetColorForText("\n\tEnter your username: ", 2);
        scanf("%s", usrnm);
        SetColorForText("\tEnter your password: ", 2);

        // password entry is kept private
        int p = 0;
        do {
            char c = getch();
            if (c == '\n') {
                break;
            }
            printf("*");
            pswd[p] = c;
            p++;
        } while (pswd[p-1] != '\r');
        pswd[p-1] = '\0';

        // first we check whether the user exists in the database
        IsUserExists = findUser(usrnm);

        // the user exists or is trying to login as the ADMIN
        if (IsUserExists || strcmp(usrnm, "admin") == 0) {
```



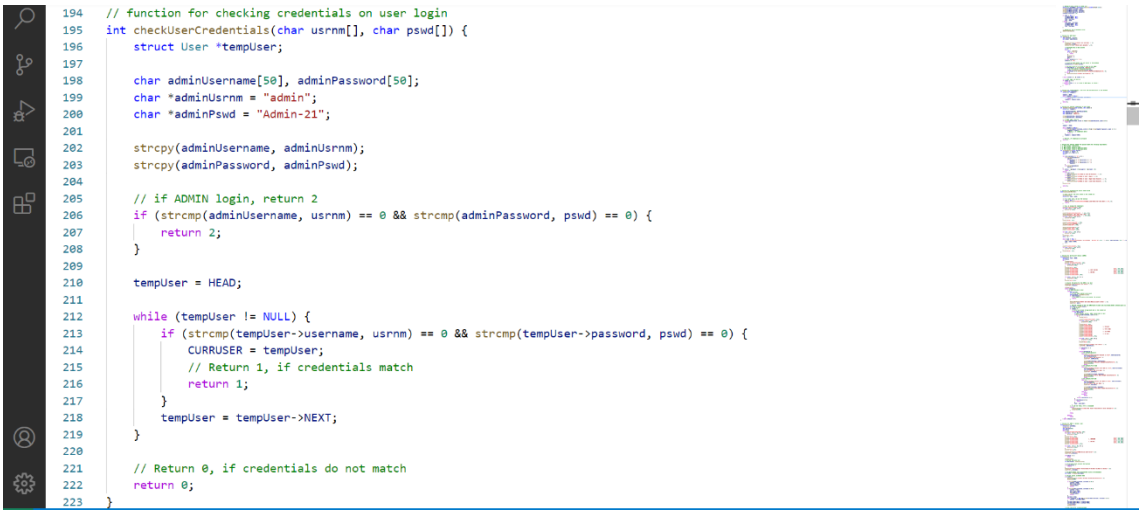
```
                break;
            }
            printf("*");
            pswd[p] = c;
            p++;
        } while (pswd[p-1] != '\r');
        pswd[p-1] = '\0';

        // first we check whether the user exists in the database
        IsUserExists = findUser(usrnm);

        // the user exists or is trying to login as the ADMIN
        if (IsUserExists || strcmp(usrnm, "admin") == 0) {
            // we check whether the password matches
            retVal = checkUserCredentials(usrnm, pswd);
            if (retVal == 0) SetColorForText("\n\tInvalid Credentials!\n", 4);
        } else {
            SetColorForText("\n\tUser Not Found!\n", 4);
        }

    } while (retVal != 1 && retVal != 2);

    // if ADMIN login, we return 2
    if (retVal == 2) {
        return 2;
    } else if (retVal == 1) { // else if USER login, we return 1
        return 1;
    }
}
```

Function: int findUser(char *UserName):



```
// function for finding whether a user with same username exists in the database
int findUser(char *username) {
    struct User *tempUser;

    tempUser = HEAD;
    while (tempUser != NULL) {
        if (!strcmp(tempUser->username, username))
            return 1;
        tempUser = tempUser->NEXT;
    }
    return 0;
}
```

Function: int checkUserCredentials(char *UserName, char *Password):



```
// function for checking credentials on user login
int checkUserCredentials(char usrnm[], char pswd[]) {
    struct User *tempUser;

    char adminUsername[50], adminPassword[50];
    char *adminUsrnm = "admin";
    char *adminPswd = "Admin-21";

    strcpy(adminUsername, adminUsrnm);
    strcpy(adminPassword, adminPswd);

    // if ADMIN login, return 2
    if (strcmp(adminUsername, usrnm) == 0 && strcmp(adminPassword, pswd) == 0) {
        return 2;
    }

    tempUser = HEAD;

    while (tempUser != NULL) {
        if (strcmp(tempUser->username, usrnm) == 0 && strcmp(tempUser->password, pswd) == 0) {
            CURRUSER = tempUser;
            // Return 1, if credentials match
            return 1;
        }
        tempUser = tempUser->NEXT;
    }

    // Return 0, if credentials do not match
    return 0;
}
```

Function: int validatePassword(char *Password)

```c
// function for checking whether the password meets the following requirements:
// 1. Must contain 8 characters
// 2. Must contain atlease 1 digit
// 3. Must contain atleast 1 uppercase letter
// 4. Must contain atleast 1 lowercase letter
int validatePassword(char *passwd) {
    int charC = 0, digitC = 0;
    int upperC = 0, lowerC = 0;
    int i;

    for (i=0; passwd[i] != '\0'; i++) {
        if (isalpha(passwd[i])){
            charC++;
            if (passwd[i] >= 'A' && passwd[i] <= 'Z')
                upperC++;
            if (passwd[i] >= 'a' && passwd[i] <= 'z')
                lowerC++;
        }
        if (isdigit(passwd[i]))
            digitC++;
    }
    if (charC > 0 && digitC > 0 && upperC > 0 && lowerC > 0)
        return 1;
    else {
        if (charC <= 8)
            SetColorForText("\n\n\tNeed at least 8 Characters...", 4);
        if (digitC == 0)
            SetColorForText("\n\tNeed at least 1 Digit...", 4);
        if (upperC == 0)
            SetColorForText("\n\tNeed at least 1 Upper Case Character...", 4);

        if (lowerC == 0)
            SetColorForText("\n\tNeed at least 1 Lower Case Character...", 4);
    }
    printf("\n");

    return 0;
}
```

Function: void displayLeaderboard()

```c
// function for displaying the points leader board
void displayLeaderBoard() {

    // First, we sort the users present in the linked list
    sortAllUsers();
    struct User *node = HEAD;

    // if no users exist, we exit the function
    if (node == NULL) {
        SetColorForText("\n\n\t\t\t\t\t\t\tSeems Like There Aren't Any Users :( \n", 4);
        return;
    }

    // Else, we display the leaderboard
    printf("\n\n\t\t\t\t\t%c", 218);
    for (int i = 0; i < 80; i++) {
        printf("%c",196);
    }

    printf("%c\n\t\t\t\t\t%c\t\t\t   ", 191, 179);
    SetColorForText(" L E A D E R B O A R D ", 2);
    printf("%-31s%c\n\t\t\t\t\t%c", " ", 179, 192);
    for (int i = 0; i < 80; i++) {
        printf("%c",196);
    }
    printf("%c\n", 217);

    printf("\t\t\t\t%c\t\t ", 179);
    SetColorForText("USERNAME", 6);
    printf("\t\t%c\t\t ", 179);

    SetColorForText("Score", 6);
    printf("\t\t\t %c\n", 179);
    printf("\t\t\t\t%c", 192);

    for (int i = 0; i < 80; i++) {
        printf("%c",196);
    }
    printf("%c", 217);
    int i = 0;

    while (node != NULL) {
        printf("\n\t\t\t\t\t%c15s%d. %s\t\t%c%15s   %d\t\t\t %c", 179, " ", (i+1), node->username, 179, " ", node->totalScore, 179);
        node = node -> NEXT;
        i++;
    }

    printf("\n\t\t\t\t\t%c", 192);
    for (int i = 0; i < 80; i++) {
        printf("%c",196);
    }
    printf("%c\n", 217);
}
```

Function: void editData()

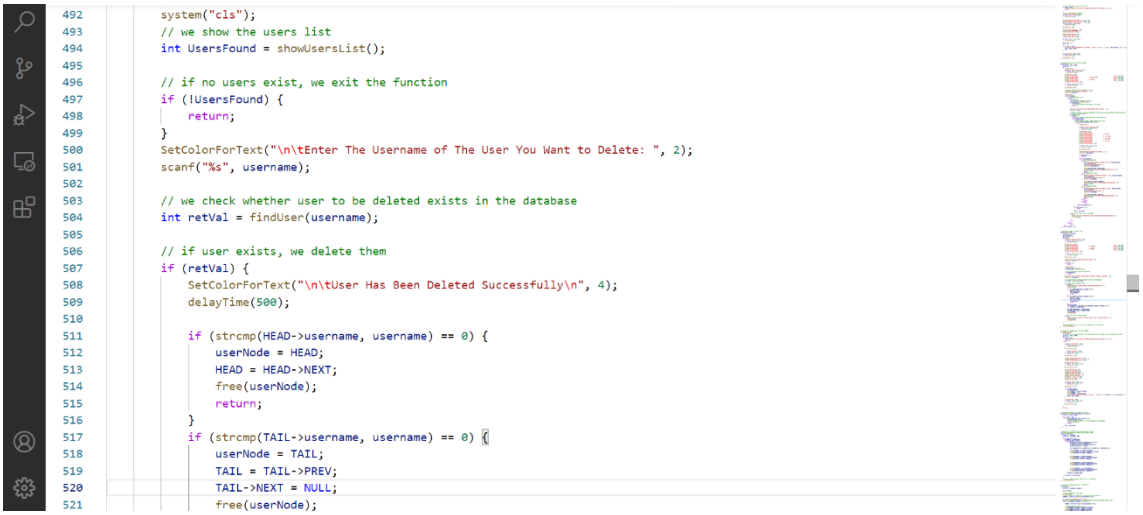
```
// function for editing user details (ADMIN)
void editData() {
    struct User *node = HEAD;
    int choice;
    do {
        system("cls");
        printf("\n\n\t\t\t\t\t\t%c", 218);
        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                1. KEEP EDITING                     %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                2. GO BACK                          %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c", 192);

        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n",217);

        // we give the option for the ADMIN to go back
        SetColorForText("\n\tEnter your Choice: ", 2);
        scanf("%d", &choice);

        system("cls");
        switch (choice) {
            // if ADMIN decides to edit
            // if ADMIN decides to edit
            case 1:;
                char user[50];
                // we first check whether users exist
                int UsersFound = showUsersList();
                if (!UsersFound) {
                    // if number of users in the system = 0, we exit
                    return;
                }

                SetColorForText("\tEnter User whose Data you want to Edit: ", 2);
                scanf("%s", user);

                // Take the username of the user ADMIN wants to edit and cross-check whether entered input is a registered user
                int retVal = findUser(user);
                // if user exists
                if (retVal) {
                    // First, we iterate through each node in the linked list
                    while (node != NULL) {
                        int editChoice;
                        // Next, we user matches, ADMIN chooses what to edit
                        if (strcmp(node->username, user) == 0) {
                            do {
                                system("cls");

                                printf("\n\n\t\t\t\t\t\t%c", 218);
                                for (int i = 0; i < 60; i++) {
                                    printf("%c",196);
                                }
                                printf("%c\n", 191);
                                printf("%c\n", 191);
                                printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                1. Password                         %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                2. First Name                       %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                3. Last Name                        %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                4. Go Back                          %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
                                printf("\t\t\t\t\t\t%c", 192);

                                for (int i = 0; i < 60; i++) {
                                    printf("%c",196);
                                }
                                printf("%c\n",217);

                                SetColorForText("\n\tEnter your choice: ", 2);
                                scanf("%d", &editChoice);

                                if (editChoice == 4)
                                    break;

                                switch (editChoice) {
                                    // for changing password
                                    case 1:;
                                        printf("\n\tYour Original Password is: %s\n", node->password);
                                        char newPassword[50];
                                        printf("\tEnter New Password: ");
                                        scanf("%s", newPassword);
```

```
                    scanf("%s", newPassword);

                    strcpy(node->password, newPassword);
                    SetColorForText("\tPassword Changed Successfully!\n", 1);
                    delayTime(2000);
                    break;
                // for changing first name
                case 2:;
                    printf("\n\tYour Original First Name is: %s\n", node->firstname);
                    char newFname[50];
                    printf("\tEnter New First Name: ");
                    scanf("%s", newFname);

                    strcpy(node->firstname, newFname);
                    SetColorForText("\tFirst Name Changed Successfully!\n", 1);
                    delayTime(2000);
                    break;
                // for changing last name
                case 3:;
                    printf("\n\tYour Original Last Name is: %s\n", node->lastname);
                    char newLname[50];
                    printf("\tEnter New Last Name: ");
                    scanf("%s", newLname);

                    strcpy(node->lastname, newLname);
                    SetColorForText("\tLast Name Changed Successfully!\n", 1);
                    delayTime(2000);
                    break;
                case 4:
                    break;
```

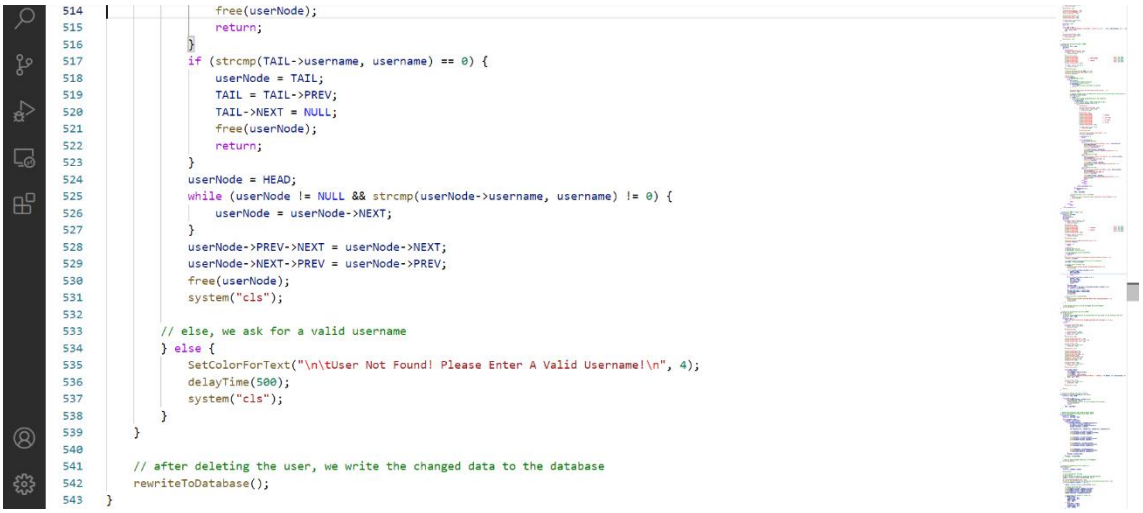
```
                    strcpy(node->lastname, newLname);
                    SetColorForText("\tLast Name Changed Successfully!\n", 1);
                    delayTime(2000);
                    break;
                case 4:
                    break;
                default:
                    break;
                }
            } while (editChoice != 4);
        }
        if (editChoice == 4) {
            break;
        }
        node = node->NEXT;
    }
    // if user not found, error is displayed
    } else {
        SetColorForText("\n\tUser Not Found! Please Enter A Valid Username!\n", 4);
        delayTime(500);
    }

    break;

default:
    break;
}
} while (choice != 2);
}
```

Function: void deleteUser()

```
// function for ADMIN to delete a user
void deleteUser() {
    struct User *userNode;
    system("cls");
    char username[50];
    int choice;
    while (1) {
        printf("\n\n\t\t\t\t\t\t%c", 218);
        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                1. CONTINUE                         %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                2. GO BACK                          %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                    %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c", 192);

        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n",217);

        SetColorForText("\n\t\tWhat do you want to do? ", 2);
        scanf("%d", &choice);

        if (choice == 2)
            break;
```

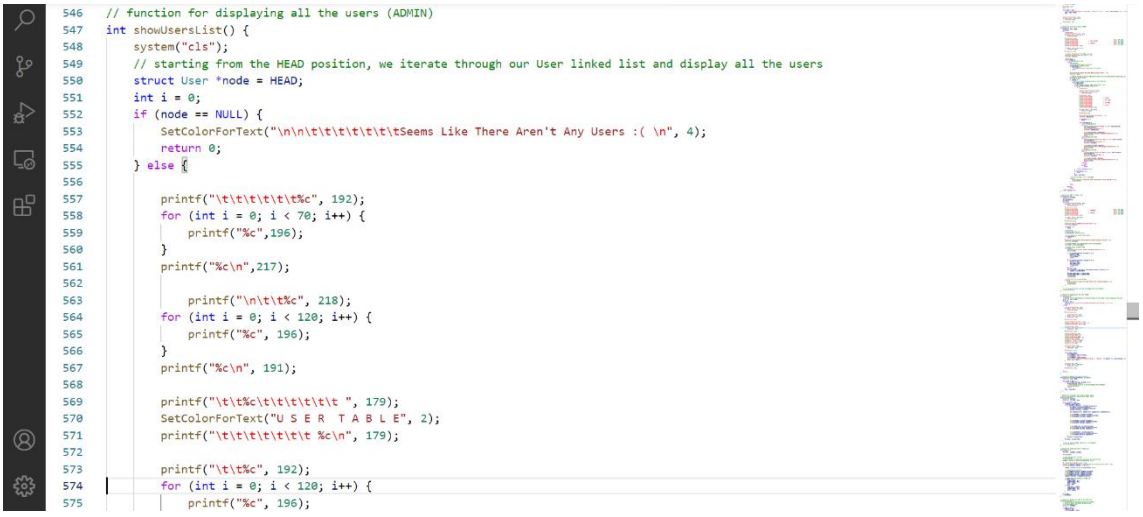


```
492         system("cls");
493         // we show the users list
494         int UsersFound = showUsersList();
495
496         // if no users exist, we exit the function
497         if (!UsersFound) {
498             return;
499         }
500         SetColorForText("\n\tEnter The Username of The User You Want to Delete: ", 2);
501         scanf("%s", username);
502
503         // we check whether user to be deleted exists in the database
504         int retVal = findUser(username);
505
506         // if user exists, we delete them
507         if (retVal) {
508             SetColorForText("\n\tUser Has Been Deleted Successfully\n", 4);
509             delayTime(500);
510
511             if (strcmp(HEAD->username, username) == 0) {
512                 userNode = HEAD;
513                 HEAD = HEAD->NEXT;
514                 free(userNode);
515                 return;
516             }
517             if (strcmp(TAIL->username, username) == 0) {
518                 userNode = TAIL;
519                 TAIL = TAIL->PREV;
520                 TAIL->NEXT = NULL;
521                 free(userNode);
```



```
514                 free(userNode);
515                 return;
516             }
517             if (strcmp(TAIL->username, username) == 0) {
518                 userNode = TAIL;
519                 TAIL = TAIL->PREV;
520                 TAIL->NEXT = NULL;
521                 free(userNode);
522                 return;
523             }
524             userNode = HEAD;
525             while (userNode != NULL && strcmp(userNode->username, username) != 0) {
526                 userNode = userNode->NEXT;
527             }
528             userNode->PREV->NEXT = userNode->NEXT;
529             userNode->NEXT->PREV = userNode->PREV;
530             free(userNode);
531             system("cls");
532
533         // else, we ask for a valid username
534         } else {
535             SetColorForText("\n\tUser Not Found! Please Enter A Valid Username!\n", 4);
536             delayTime(500);
537             system("cls");
538         }
539     }
540
541     // after deleting the user, we write the changed data to the database
542     rewriteToDatabase();
543 }
```

Function: int showUsersList()



```
546 // function for displaying all the users (ADMIN)
547 int showUsersList() {
548     system("cls");
549     // starting from the HEAD position, we iterate through our User linked list and display all the users
550     struct User *node = HEAD;
551     int i = 0;
552     if (node == NULL) {
553         SetColorForText("\n\n\t\t\t\t\t\t\tSeems Like There Aren't Any Users :( \n", 4);
554         return 0;
555     } else {
556
557         printf("\t\t\t\t\t\t%c", 192);
558         for (int i = 0; i < 70; i++) {
559             printf("%c",196);
560         }
561         printf("%c\n",217);
562
563             printf("\n\t\t%c", 218);
564         for (int i = 0; i < 120; i++) {
565             printf("%c", 196);
566         }
567         printf("%c\n", 191);
568
569         printf("\t\t%c\t\t\t\t\t\t\t ", 179);
570         SetColorForText("U S E R  T A B L E", 2);
571         printf("\t\t\t\t\t\t\t %c\n", 179);
572
573         printf("\t\t%c", 192);
574         for (int i = 0; i < 120; i++) {
575             printf("%c", 196);
```

```
574         for (int i = 0; i < 120; i++) {
575             printf("%c", 196);
576         }
577         printf("%c\n", 217);
578 
579         printf("\t\t%c\t\t", 179);
580         SetColorForText("Name", 6);
581         printf("\t\t%c\t\t", 179);
582         SetColorForText("User Name", 6);
583         printf("\t\t%c\t", 179);
584         SetColorForText("Password", 6);
585         printf("\t    %c\t   ", 179);
586         SetColorForText("Score", 6);
587         printf("\t %c", 179);
588 
589         printf("\n\t\t%c", 192);
590         for (int i = 0; i < 120; i++) {
591             printf("%c", 196);
592         }
593         printf("%c\n", 217);
594 
595         while (node != NULL) {
596             char NameStr[60];
597             strcpy(NameStr, node->firstname);
598             strcat(NameStr, " ");
599             strcat(NameStr, node->lastname);
600             printf("\t\t%c\t%-24s%c\t\t%-24s%c\t%-20s%c\t   %-14d%c\n", 179, NameStr, 179, node->username, 179, node->password, 179, node->
601             node = node -> NEXT;
602         }
603 
604         printf("\t\t%c", 192);
605         for (int i = 0; i < 120; i++) {
606             printf("%c", 196);
607         }
608         printf("%c\n", 217);
609 
610     }
611     return 1;
612 }
```

Function: void updateUserScore(char *UserName, int Score)

```
615 // function for updating the score of the user
616 void updateUserScore(char username[50], int Score) {
617     struct User *node = HEAD;
618 
619     while (node != NULL) {
620         if (strcmp(node->username, username) == 0) {
621             node->totalScore = Score;
622             // after setting the score, we rewrite changed data to database
623             rewriteToDatabase();
624             return;
625         }
626         node = node->NEXT;
627     }
628 }
```

Function: void sortAllUsers()

```
631 // function for sorting all users based on points scored
632 // done for developing the ranking system (leader board)
633 void sortAllUsers() {
634     struct User *nextNode;
635     struct User *currnode = HEAD;
636 
637     while (currnode != NULL) {
638         nextNode = currnode->NEXT;
639         while (nextNode != NULL) {
640             if (currnode->totalScore < nextNode->totalScore) {
641                 int tempInt = currnode->totalScore;
642                 currnode->totalScore = nextNode->totalScore;
643                 nextNode->totalScore = tempInt;
644 
645                 char tempUserStr[30], tempFNStr[30], tempLNStr[30], tempPswdStr[30];
646 
647                 strcpy(tempFNStr, currnode->firstname);
648                 strcpy(currnode->firstname, nextNode->firstname);
649                 strcpy(nextNode->firstname, tempFNStr);
650 
651 
652                 strcpy(tempLNStr, currnode->lastname);
653                 strcpy(currnode->lastname, nextNode->lastname);
654                 strcpy(nextNode->lastname, tempLNStr);
655 
656 
657                 strcpy(tempUserStr, currnode->username);
658                 strcpy(currnode->username, nextNode->username);
659                 strcpy(nextNode->username, tempUserStr);
```

```
                strcpy(tempPswdStr, currnode->password);
                strcpy(currnode->password, nextNode->password);
                strcpy(nextNode->password, tempPswdStr);
            }
            nextNode = nextNode->NEXT;
        }
        currnode = currnode->NEXT;
    }

    // Then, we rewrite changed linked list to file database
    rewriteToDatabase();
}
```

Function: void loadUsers()

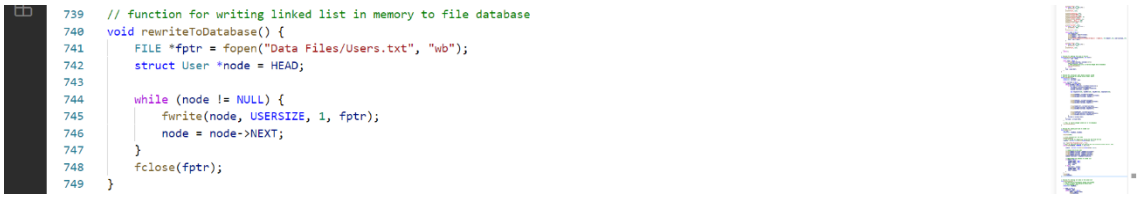
```
// function for loading user data to linked list
void loadUsers() {
    FILE *fp;
    struct User *tempNode, *newNode;

    delayTime(500);

    // First, we delete all the nodes
    deleteAllNodes();
    // this is a node for temporarily storing the data from the file
    tempNode = (struct User*)malloc(sizeof(struct User));

    fp = fopen("Data Files/Users.txt", "rb");
    // Next, we read from the file such that we read each structure/record one at a time.
    while(fread(tempNode, USERSIZE, 1, fp) > 0){

        newNode = (struct User*)malloc(sizeof(struct User));

        // copying data to new node
        strcpy(newNode->firstname, tempNode->firstname);
        strcpy(newNode->lastname, tempNode->lastname);
        strcpy(newNode->username, tempNode->username);
        strcpy(newNode->password, tempNode->password);
        newNode->totalScore = tempNode->totalScore;

        // adding data from database to linked list
        if (HEAD == NULL) {
            newNode->PREV = NULL;
            newNode->NEXT = NULL;
```

```
        // adding data from database to linked list
        if (HEAD == NULL) {
            newNode->PREV = NULL;
            newNode->NEXT = NULL;
            HEAD = newNode;
            TAIL = HEAD;
        } else {
            TAIL->NEXT = newNode;
            newNode->PREV = TAIL;
            newNode->NEXT = NULL;
            TAIL = newNode;
        }
    }
    fclose(fp);
    free(tempNode);
}
```

Function: void deleteAllNodes()

```
// function for deleting all nodes in the linked list
void deleteAllNodes() {
    // this function is utilized for wiping out old data
    // present in memory and adding new data present
    // in the database (file)
    struct User *tempNode;

    if (HEAD != NULL) {
        tempNode = HEAD;
        while (tempNode != NULL){
            HEAD = tempNode->NEXT;
            free(tempNode);
            tempNode = HEAD;
        }
        HEAD = NULL;
        TAIL = NULL;
    }
}
```

Function: void rewriteToDatabase()

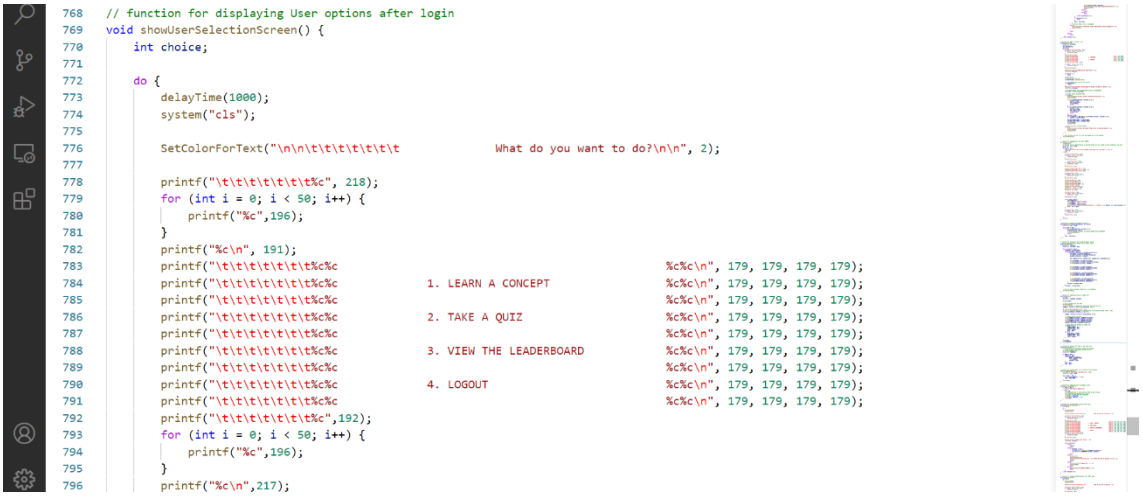


```c
// function for writing linked list in memory to file database
void rewriteToDatabase() {
    FILE *fptr = fopen("Data Files/Users.txt", "wb");
    struct User *node = HEAD;

    while (node != NULL) {
        fwrite(node, USERSIZE, 1, fptr);
        node = node->NEXT;
    }
    fclose(fptr);
}
```

Function: void addUserToDatabase()

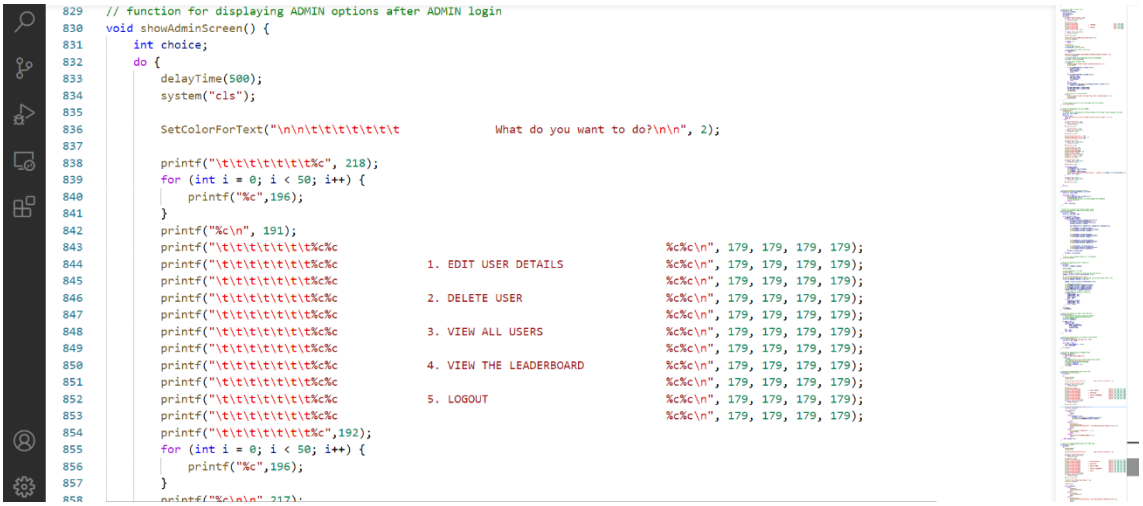
```c
// function for adding new user to database (file)
void addUserToDatabase() {
    if (TAIL == NULL) {
        printf("User Table is empty!\n");
    } else {
        FILE *fp;
        // adding user data to file in binary form for user privacy
        fp = fopen("Data Files/Users.txt", "ab");
        // we seek till EOF and add to database
        fseek(fp, 0, SEEK_END);
        fwrite(TAIL, USERSIZE, 1, fp);
        fclose(fp);
    }
}
```

Function: void showUserSelectionScreen()



```c
// function for displaying User options after login
void showUserSelectionScreen() {
    int choice;

    do {
        delayTime(1000);
        system("cls");

        SetColorForText("\n\n\t\t\t\t\t\t\t\t             What do you want to do?\n\n", 2);

        printf("\t\t\t\t\t\t\t%c", 218);
        for (int i = 0; i < 50; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t\t\t%c%c                                                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             1. LEARN A CONCEPT                   %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             2. TAKE A QUIZ                       %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             3. VIEW THE LEADERBOARD              %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             4. LOGOUT                            %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c",192);
        for (int i = 0; i < 50; i++) {
            printf("%c",196);
        }
        printf("%c\n",217);

        SetColorForText("\tEnter your choice: ", 2);
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                learn();
                break;
            case 2:;
                if (CURRUSER != NULL) {
                    int newScore = takeMainQuiz(CURRUSER->totalScore);
                    updateUserScore(CURRUSER->username, newScore);
                }
                break;
            case 3:
                system("cls");
                displayLeaderBoard();
                SetColorForText("\n\t\t\t\t\t\t\t\t   <<<<< Press Any Key To Continue >>>>>\n", 2);
                getch();
                break;
            case 4:
                SetColorForText("\tLogging out....", 4);
                delayTime(500);
                break;
            default:
                SetColorForText("\tINVALID ENTRY!\n", 4);
                break;
        }
    } while (choice != 4);
}
```

Function: void showAdminScreen()

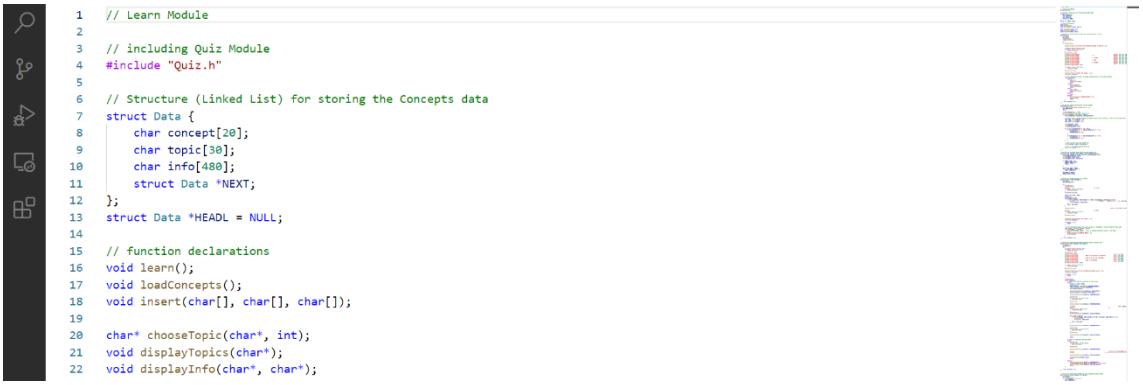


```
// function for displaying ADMIN options after ADMIN login
void showAdminScreen() {
    int choice;
    do {
        delayTime(500);
        system("cls");

        SetColorForText("\n\n\t\t\t\t\t\t\t\t             What do you want to do?\n\n", 2);

        printf("\t\t\t\t\t\t\t%c", 218);
        for (int i = 0; i < 50; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             1. EDIT USER DETAILS               %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             2. DELETE USER                     %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             3. VIEW ALL USERS                  %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             4. VIEW THE LEADERBOARD            %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c             5. LOGOUT                          %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c%c                                                %c%c\n", 179, 179, 179, 179);
        printf("\t\t\t\t\t\t\t%c",192);
        for (int i = 0; i < 50; i++) {
            printf("%c",196);
        }
        printf("%c\n\n",217);

        SetColorForText("\tEnter your choice: ", 2);
        scanf("%d", &choice);

        system("cls");

        switch (choice) {
            case 1:
                editData();
                rewriteToDatabase();
                break;
            case 2:
                deleteUser();
                rewriteToDatabase();
                break;
            case 3:
                showUsersList();
                SetColorForText("\n\t\t\t\t\t\t\t  <<<<< Press Any Key To Continue >>>>>\n", 2);
                getch();
                break;
            case 4:
                displayLeaderBoard();
                SetColorForText("\n\t\t\t\t\t\t\t  <<<<< Press Any Key To Continue >>>>>\n", 2);
                getch();
                break;
            case 5:
                SetColorForText("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\tLogging out....", 4);
                delayTime(500);
                break;
            default: printf("\tINVALID ENTRY!\n");
                break;
        }
        if (choice == 5)
            break;
    } while (choice != 5);
}
```

3.2.1.2. LEARN

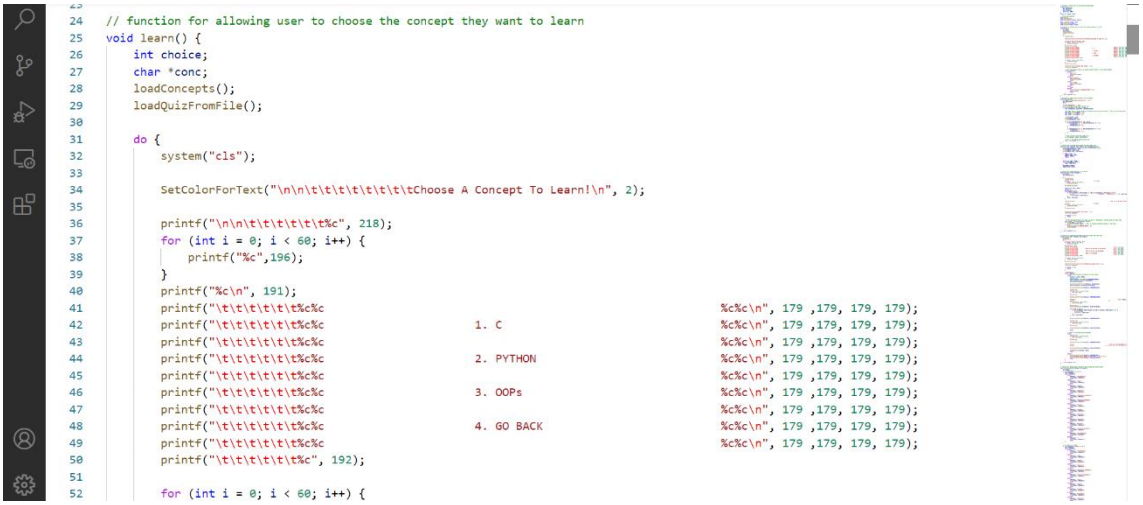
The Learn Module contains the functions related to displaying the information regarding the chosen concept and topic.

Function declarations and Linked List structure

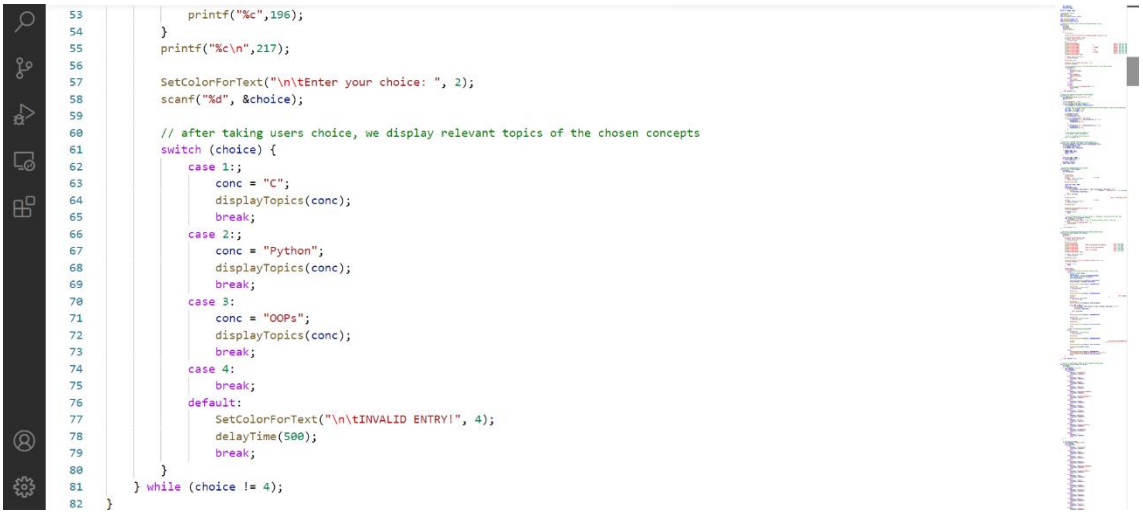


```c
// Learn Module

// including Quiz Module
#include "Quiz.h"

// Structure (Linked List) for storing the Concepts data
struct Data {
    char concept[20];
    char topic[30];
    char info[480];
    struct Data *NEXT;
};
struct Data *HEADL = NULL;

// function declarations
void learn();
void loadConcepts();
void insert(char[], char[], char[]);

char* chooseTopic(char*, int);
void displayTopics(char*);
void displayInfo(char*, char*);
```

Function: void learn()

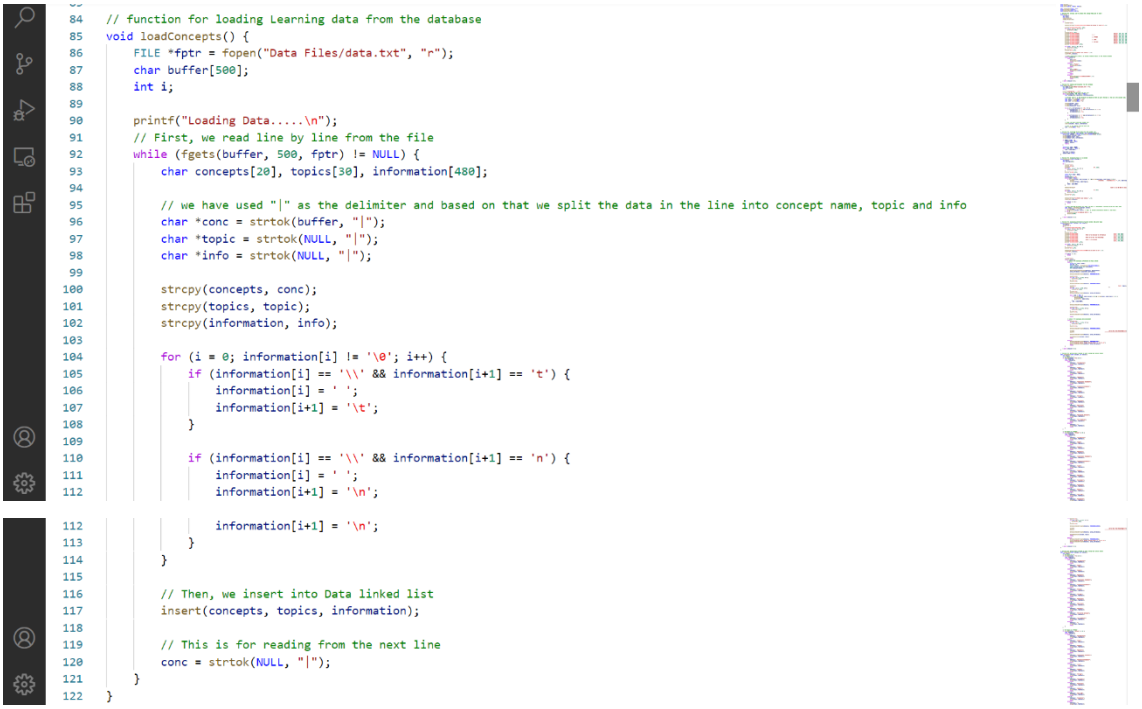


```c
// function for allowing user to choose the concept they want to learn
void learn() {
    int choice;
    char *conc;
    loadConcepts();
    loadQuizFromFile();

    do {
        system("cls");

        SetColorForText("\n\n\t\t\t\t\t\t\t\tChoose A Concept To Learn!\n", 2);

        printf("\n\n\t\t\t\t\t%c", 218);
        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t%c%c                                                  %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                    1. C                          %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                                                  %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                    2. PYTHON                     %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                                                  %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                    3. OOPs                       %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                                                  %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                    4. GO BACK                    %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c%c                                                  %c%c\n", 179 ,179, 179, 179);
        printf("\t\t\t\t\t%c", 192);

        for (int i = 0; i < 60; i++) {
```



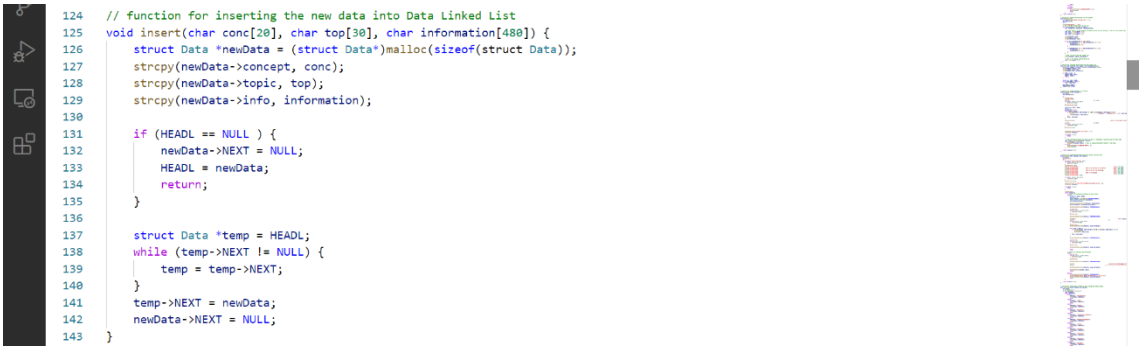
```c
            printf("%c",196);
        }
        printf("%c\n",217);

        SetColorForText("\n\tEnter your choice: ", 2);
        scanf("%d", &choice);

        // after taking users choice, we display relevant topics of the chosen concepts
        switch (choice) {
            case 1:;
                conc = "C";
                displayTopics(conc);
                break;
            case 2:;
                conc = "Python";
                displayTopics(conc);
                break;
            case 3:
                conc = "OOPs";
                displayTopics(conc);
                break;
            case 4:
                break;
            default:
                SetColorForText("\n\tINVALID ENTRY!", 4);
                delayTime(500);
                break;
        }
    } while (choice != 4);
}
```

Function: void loadConcepts()



```c
// function for loading Learning data from the database
void loadConcepts() {
    FILE *fptr = fopen("Data Files/data.txt", "r");
    char buffer[500];
    int i;

    printf("Loading Data.....\n");
    // First, we read line by line from the file
    while (fgets(buffer, 500, fptr) != NULL) {
        char concepts[20], topics[30], information[480];

        // we have used "|" as the delimiter and based on that we split the data in the line into concept name, topic and info
        char *conc = strtok(buffer, "|");
        char *topic = strtok(NULL, "|");
        char *info = strtok(NULL, "|");

        strcpy(concepts, conc);
        strcpy(topics, topic);
        strcpy(information, info);

        for (i = 0; information[i] != '\0'; i++) {
            if (information[i] == '\\' && information[i+1] == 't') {
                information[i] = ' ';
                information[i+1] = '\t';
            }

            if (information[i] == '\\' && information[i+1] == 'n') {
                information[i] = ' ';
                information[i+1] = '\n';
                information[i+1] = '\n';
            }
        }

        // Then, we insert into Data linked list
        insert(concepts, topics, information);

        // This is for reading from the next line
        conc = strtok(NULL, "|");
    }
}
```

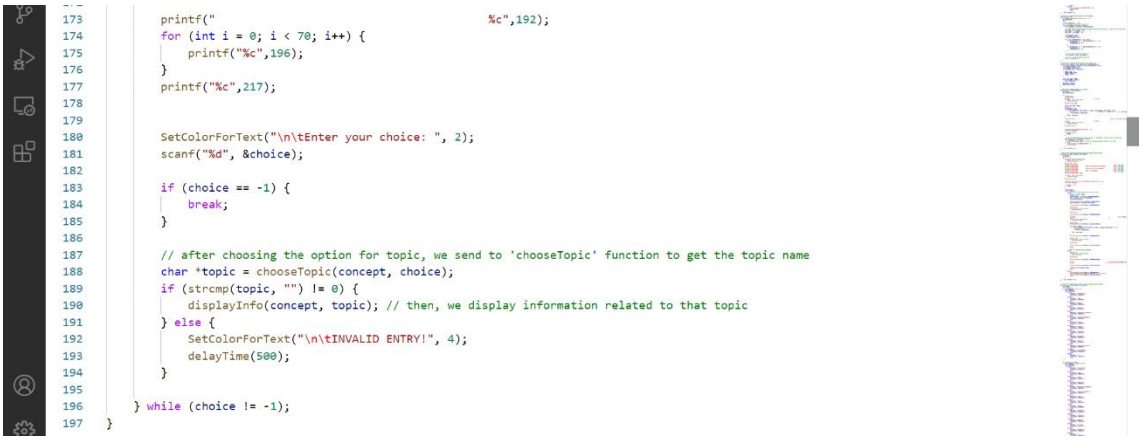
Function: void insert(char *Concept, char *Topic, char *Information)



```c
// function for inserting the new data into Data Linked List
void insert(char conc[20], char top[30], char information[480]) {
    struct Data *newData = (struct Data*)malloc(sizeof(struct Data));
    strcpy(newData->concept, conc);
    strcpy(newData->topic, top);
    strcpy(newData->info, information);

    if (HEADL == NULL ) {
        newData->NEXT = NULL;
        HEADL = newData;
        return;
    }

    struct Data *temp = HEADL;
    while (temp->NEXT != NULL) {
        temp = temp->NEXT;
    }
    temp->NEXT = newData;
    newData->NEXT = NULL;
}
```

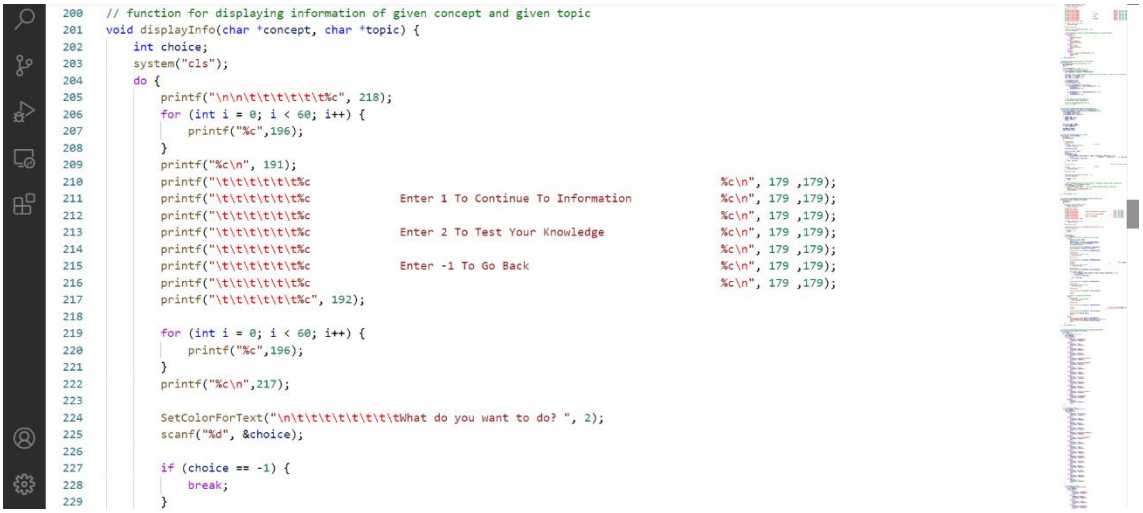
Function: void displayTopics(char *Concept)

```
// function for displaying topics in a concept
void displayTopics(char *concept) {
    int choice;
    char prevTopic[30];

    do {
        system("cls");
        printf("\n\n");
        printf("                                        %c", 218);
        for (int i = 0; i < 70; i++) {
            printf("%c",196);
        }
        printf("%c\n\n",191);

        struct Data *node = HEADL;
        int i = 1;
        prevTopic[0] = '\0';
        while (node != NULL) {
            if (strcmp(concept, node->concept) == 0 && strcmp(prevTopic, node->topic) != 0) {
                printf("                                        |%-23s%2d.    %-22s%18s\n\n"," ", i++, node->topic,"|");
                strcpy(prevTopic, node->topic);
            }
            node = node->NEXT;
        }

        SetColorForText("                                                                Enter -1 to GO BACK to Concepts\n\n", 2);
```



```
        printf("                                        %c",192);
        for (int i = 0; i < 70; i++) {
            printf("%c",196);
        }
        printf("%c",217);


        SetColorForText("\n\tEnter your choice: ", 2);
        scanf("%d", &choice);

        if (choice == -1) {
            break;
        }

        // after choosing the option for topic, we send to 'chooseTopic' function to get the topic name
        char *topic = chooseTopic(concept, choice);
        if (strcmp(topic, "") != 0) {
            displayInfo(concept, topic); // then, we display information related to that topic
        } else {
            SetColorForText("\n\tINVALID ENTRY!", 4);
            delayTime(500);
        }

    } while (choice != -1);
}
```

Function: void displayInfo(char *Concept, char *Topic)



```
// function for displaying information of given concept and given topic
void displayInfo(char *concept, char *topic) {
    int choice;
    system("cls");
    do {
        printf("\n\n\t\t\t\t\t\t%c", 218);
        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n", 191);
        printf("\t\t\t\t\t\t%c                                                %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c            Enter 1 To Continue To Information  %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c            Enter 2 To Test Your Knowledge      %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c            Enter -1 To Go Back                 %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c                                                %c\n", 179 ,179);
        printf("\t\t\t\t\t\t%c", 192);

        for (int i = 0; i < 60; i++) {
            printf("%c",196);
        }
        printf("%c\n",217);

        SetColorForText("\n\t\t\t\t\t\t\t\tWhat do you want to do? ", 2);
        scanf("%d", &choice);

        if (choice == -1) {
            break;
        }
```

```
231         system("cls");
232         switch (choice) {
233             // choice for displaying information on topic chosen
234             case 1:;
235                 struct Data *node = HEADL;
236                 int iter = 0;
237                 HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
238                 CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
239                 WORD saved_attributes;
240 
241                 GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
242                 saved_attributes = consoleInfo.wAttributes;
243 
244                 SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
245 
246                 printf("\n");
247                 for (int i = 0; i < 170; i++) {
248                     printf("%c",196);
249                 }
250                 printf("\n");
251 
252                 SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
253 
254                 printf("\n                                                                    %s\n", topic);
255                 printf("                                                          ");
256                 for (int i = 0; i < 40; i++) {
257                     printf("%c",196);
258                 }
259                 printf("\n");
```

```
259                 printf("\n");
260                 SetConsoleTextAttribute(hConsole, saved_attributes);
261 
262                 while (node != NULL) {
263                     if (strcmp(concept, node->concept) == 0 && strcmp(topic, node->topic) == 0) {
264                         delayTime(100);
265                         printf("%s", node->info);
266                     }
267                     node = node->NEXT;
268                 }
269 
270                 SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
271 
272                 printf("\n");
273                 for (int i = 0; i < 170; i++) {
274                     printf("%c",196);
275                 }
276                 printf("\n");
277 
278                 SetConsoleTextAttribute(hConsole, saved_attributes);
279 
280                 break;
281 
282             // choice for displaying mini-assessment
283             case 2:
284                 printf("\n");
285                 for (int i = 0; i < 170; i++) {
286                     printf("%c",196);
287                 }
288                 printf("\n");
```

```
290                 SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
291 
292                 printf("                                                    Let's Test Your Knowledge!\n");
293                 printf("                                                ----------------------------------------\n");
294 
295                 SetConsoleTextAttribute(hConsole, saved_attributes);
296 
297                 displayQuestions(concept, topic);
298                 break;
299 
300             default:
301                 SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
302                 printf("Invalid Choice. Please Choose Either -1, 1 or 2.\n");
303                 SetConsoleTextAttribute(hConsole, saved_attributes);
304                 break;
305         }
306 
307     } while (choice != -1);
308 }
```

Function: void chooseTopic(char *Concept, int optionChosen)

```c
// function for getting topic related to given concept and option chosen
char* chooseTopic(char* concept, int choice) {
    char *topic;
    // for topics in 'C'
    if (strcmp(concept, "C") == 0) {
        char *tempTopic;
        switch (choice) {
            case 1:
                tempTopic = "Introduction";
                strcpy(topic, tempTopic);
                break;
            case 2:
                tempTopic = "Input";
                strcpy(topic, tempTopic);
                break;
            case 3:
                tempTopic = "Output";
                strcpy(topic, tempTopic);
                break;
            case 4:
                tempTopic = "Operators";
                strcpy(topic, tempTopic);
                break;
            case 5:
                tempTopic = "Conditional Statements";
                strcpy(topic, tempTopic);
                break;
            case 6:
                tempTopic = "Iterative Statements";
                strcpy(topic, tempTopic);
```

```c
            case 6:
                tempTopic = "Iterative Statements";
                strcpy(topic, tempTopic);
                break;
            case 7:
                tempTopic = "Arrays";
                strcpy(topic, tempTopic);
                break;
            case 8:
                tempTopic = "Strings";
                strcpy(topic, tempTopic);
                break;
            case 9:
                tempTopic = "Functions";
                strcpy(topic, tempTopic);
                break;
            case 10:
                tempTopic = "Recursion";
                strcpy(topic, tempTopic);
                break;
            case 11:
                tempTopic = "Pointers";
                strcpy(topic, tempTopic);
                break;
            case 12:
                tempTopic = "Structures & Unions";
                strcpy(topic, tempTopic);
                break;
            case 13:
                tempTopic = "File Handling";
```

```c
            case 13:
                tempTopic = "File Handling";
                strcpy(topic, tempTopic);
                break;
            default:
                tempTopic = "";
                strcpy(topic, tempTopic);
                break;
        }
    }

    // for topics in 'Python'
    if (strcmp(concept, "Python") == 0) {
        char *tempTopic;
        switch (choice) {
            case 1:
                tempTopic = "Introduction";
                strcpy(topic, tempTopic);
                break;
            case 2:
                tempTopic = "Input";
                strcpy(topic, tempTopic);
                break;
            case 3:
                tempTopic = "Output";
                strcpy(topic, tempTopic);
                break;
            case 4:
                tempTopic = "Operators";
                strcpy(topic, tempTopic);
```

```
            case 4:
                tempTopic = "Operators";
                strcpy(topic, tempTopic);
                break;
            case 5:
                tempTopic = "Conditional Statements";
                strcpy(topic, tempTopic);
                break;
            case 6:
                tempTopic = "Iterative Statements";
                strcpy(topic, tempTopic);
                break;
            case 7:
                tempTopic = "Lists";
                strcpy(topic, tempTopic);
                break;
            case 8:
                tempTopic = "Tuples";
                strcpy(topic, tempTopic);
                break;
            case 9:
                tempTopic = "Strings";
                strcpy(topic, tempTopic);
                break;
            case 10:
                tempTopic = "Dictionary";
                strcpy(topic, tempTopic);
                break;
            case 11:
                tempTopic = "Functions";
```

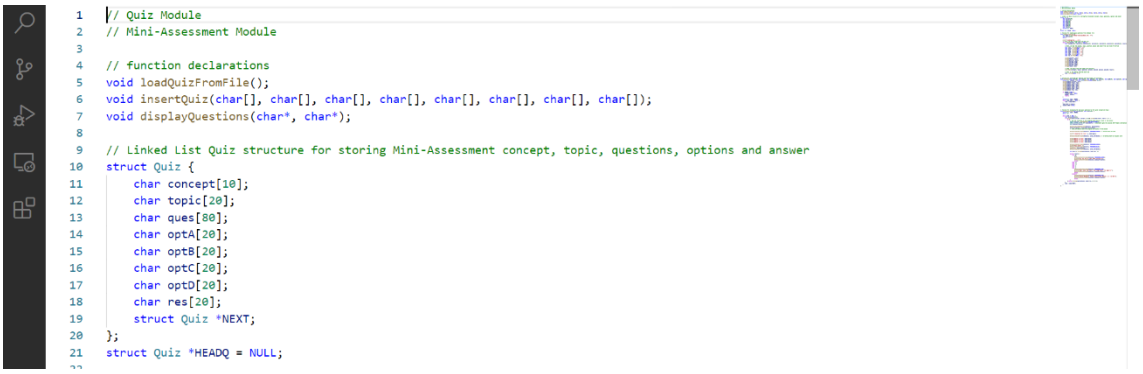
```
                strcpy(topic, tempTopic);
                break;
            case 12:
                tempTopic = "Modules";
                strcpy(topic, tempTopic);
                break;
            case 13:
                tempTopic = "File I/O";
                strcpy(topic, tempTopic);
                break;
            case 14:
                tempTopic = "Exceptions";
                strcpy(topic, tempTopic);
                break;
            default:
                tempTopic = "";
                strcpy(topic, tempTopic);
                break;
        }

        // for topics in 'OOPs'
        if (strcmp(concept, "OOPs") == 0) {
            char *tempTopic;
            switch (choice) {
                case 1:
                    tempTopic = "Introduction";
                    strcpy(topic, tempTopic);
                    break;
                case 2:
                    tempTopic = "Objects";
```

```
                    break;
                case 3:
                    tempTopic = "Classes";
                    strcpy(topic, tempTopic);
                    break;
                case 4:
                    tempTopic = "Inheritance";
                    strcpy(topic, tempTopic);
                    break;
                case 5:
                    tempTopic = "Polymorphism";
                    strcpy(topic, tempTopic);
                    break;
                case 6:
                    tempTopic = "Abstraction";
                    strcpy(topic, tempTopic);
                    break;
                case 7:
                    tempTopic = "Encaspulation";
                    strcpy(topic, tempTopic);
                    break;
                default:
                    tempTopic = "";
                    strcpy(topic, tempTopic);
                    break;
            }
        }
    }
    return topic;
}
```

3.2.1.3. MINI-ASSESSMENT

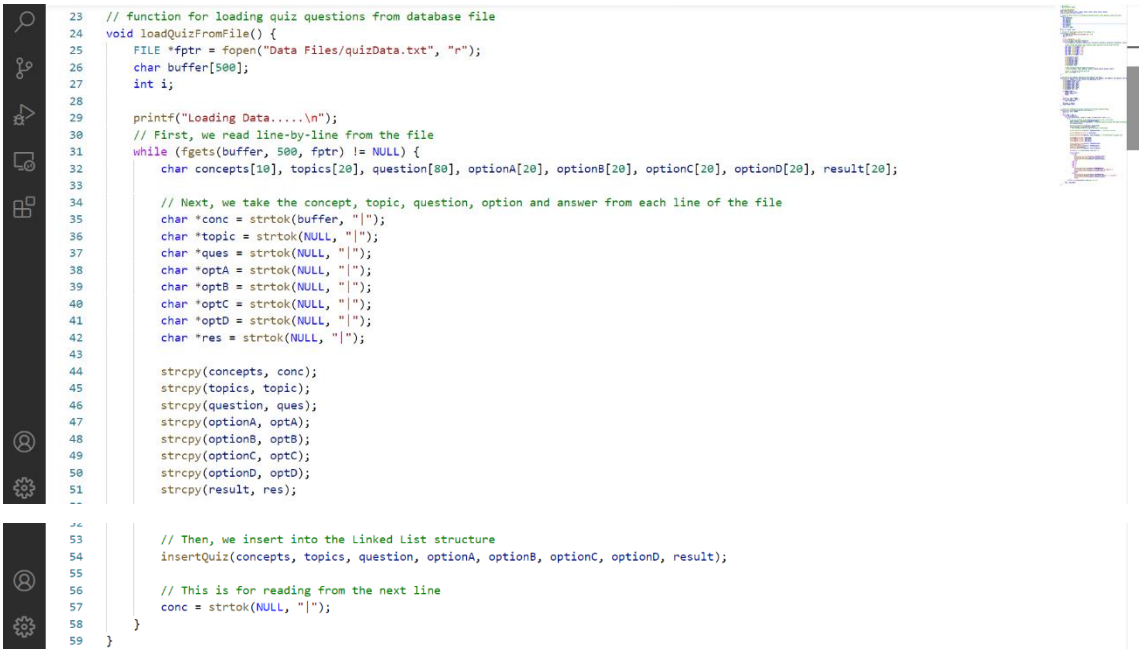
The Mini-Assessment module (named Quiz.h) is an important component in CODIAC which contains the self-assessment section within the Learn module.

Function declarations and Quiz Linked List Structure



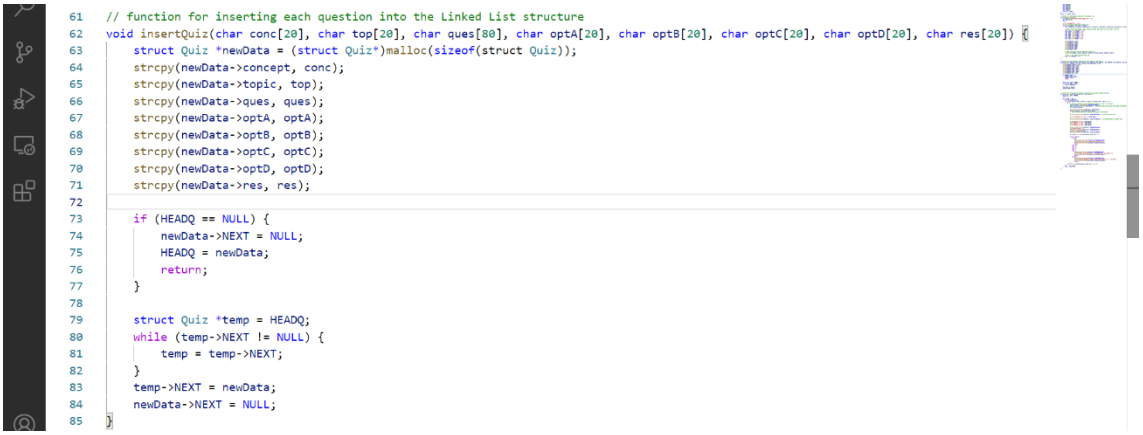
```c
// Quiz Module
// Mini-Assessment Module

// function declarations
void loadQuizFromFile();
void insertQuiz(char[], char[], char[], char[], char[], char[], char[], char[]);
void displayQuestions(char*, char*);

// Linked List Quiz structure for storing Mini-Assessment concept, topic, questions, options and answer
struct Quiz {
    char concept[10];
    char topic[20];
    char ques[80];
    char optA[20];
    char optB[20];
    char optC[20];
    char optD[20];
    char res[20];
    struct Quiz *NEXT;
};
struct Quiz *HEADQ = NULL;
```

Function: void loadQuizFromFile()



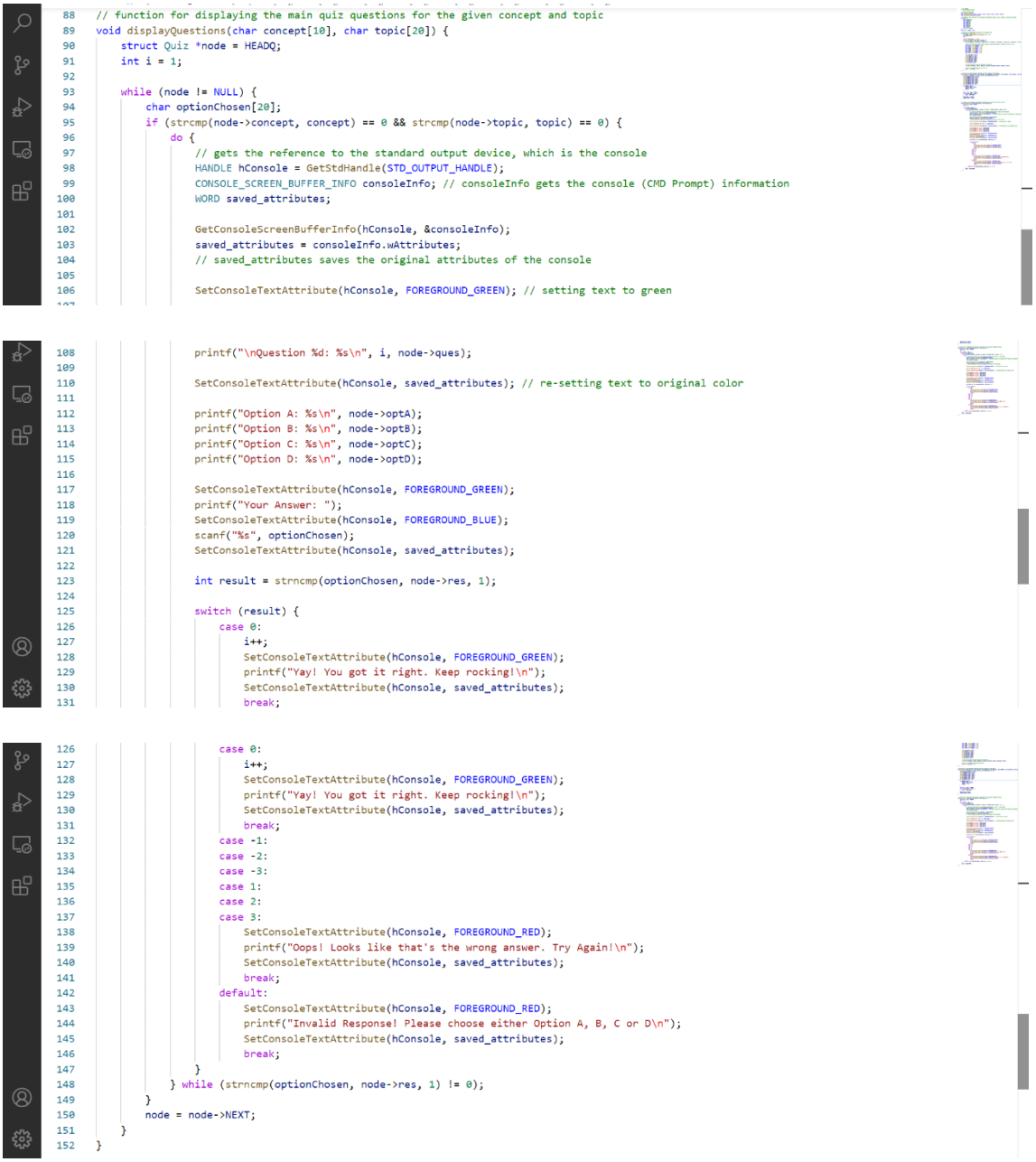
```c
// function for loading quiz questions from database file
void loadQuizFromFile() {
    FILE *fptr = fopen("Data Files/quizData.txt", "r");
    char buffer[500];
    int i;

    printf("Loading Data.....\n");
    // First, we read line-by-line from the file
    while (fgets(buffer, 500, fptr) != NULL) {
        char concepts[10], topics[20], question[80], optionA[20], optionB[20], optionC[20], optionD[20], result[20];

        // Next, we take the concept, topic, question, option and answer from each line of the file
        char *conc = strtok(buffer, "|");
        char *topic = strtok(NULL, "|");
        char *ques = strtok(NULL, "|");
        char *optA = strtok(NULL, "|");
        char *optB = strtok(NULL, "|");
        char *optC = strtok(NULL, "|");
        char *optD = strtok(NULL, "|");
        char *res = strtok(NULL, "|");

        strcpy(concepts, conc);
        strcpy(topics, topic);
        strcpy(question, ques);
        strcpy(optionA, optA);
        strcpy(optionB, optB);
        strcpy(optionC, optC);
        strcpy(optionD, optD);
        strcpy(result, res);

        // Then, we insert into the Linked List structure
        insertQuiz(concepts, topics, question, optionA, optionB, optionC, optionD, result);

        // This is for reading from the next line
        conc = strtok(NULL, "|");
    }
}
```

Function: void insertQuiz(char *Concept, char *Topic, char *Question, char *OptA, char *OptB, char *OptC, char *OptD, char *Res)



```c
// function for inserting each question into the Linked List structure
void insertQuiz(char conc[20], char top[20], char ques[80], char optA[20], char optB[20], char optC[20], char optD[20], char res[20]) {
    struct Quiz *newData = (struct Quiz*)malloc(sizeof(struct Quiz));
    strcpy(newData->concept, conc);
    strcpy(newData->topic, top);
    strcpy(newData->ques, ques);
    strcpy(newData->optA, optA);
    strcpy(newData->optB, optB);
    strcpy(newData->optC, optC);
    strcpy(newData->optD, optD);
    strcpy(newData->res, res);

    if (HEADQ == NULL) {
        newData->NEXT = NULL;
        HEADQ = newData;
        return;
    }

    struct Quiz *temp = HEADQ;
    while (temp->NEXT != NULL) {
        temp = temp->NEXT;
    }
    temp->NEXT = newData;
    newData->NEXT = NULL;
}
```

Function: void displayQuestions(char *Concept, char *Topic)



```c
// function for displaying the main quiz questions for the given concept and topic
void displayQuestions(char concept[10], char topic[20]) {
    struct Quiz *node = HEADQ;
    int i = 1;

    while (node != NULL) {
        char optionChosen[20];
        if (strcmp(node->concept, concept) == 0 && strcmp(node->topic, topic) == 0) {
            do {
                // gets the reference to the standard output device, which is the console
                HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                CONSOLE_SCREEN_BUFFER_INFO consoleInfo; // consoleInfo gets the console (CMD Prompt) information
                WORD saved_attributes;

                GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
                saved_attributes = consoleInfo.wAttributes;
                // saved_attributes saves the original attributes of the console

                SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN); // setting text to green

                printf("\nQuestion %d: %s\n", i, node->ques);

                SetConsoleTextAttribute(hConsole, saved_attributes); // re-setting text to original color

                printf("Option A: %s\n", node->optA);
                printf("Option B: %s\n", node->optB);
                printf("Option C: %s\n", node->optC);
                printf("Option D: %s\n", node->optD);

                SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
                printf("Your Answer: ");
                SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
                scanf("%s", optionChosen);
                SetConsoleTextAttribute(hConsole, saved_attributes);

                int result = strncmp(optionChosen, node->res, 1);

                switch (result) {
                    case 0:
                        i++;
                        SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
                        printf("Yay! You got it right. Keep rocking!\n");
                        SetConsoleTextAttribute(hConsole, saved_attributes);
                        break;
                    case -1:
                    case -2:
                    case -3:
                    case 1:
                    case 2:
                    case 3:
                        SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
                        printf("Oops! Looks like that's the wrong answer. Try Again!\n");
                        SetConsoleTextAttribute(hConsole, saved_attributes);
                        break;
                    default:
                        SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
                        printf("Invalid Response! Please choose either Option A, B, C or D\n");
                        SetConsoleTextAttribute(hConsole, saved_attributes);
                        break;
                }
            } while (strncmp(optionChosen, node->res, 1) != 0);
        }
        node = node->NEXT;
    }
}
```

3.2.1.4. MAIN QUIZ

The MainQuiz module contains the functions regarding displaying the MainQuiz questions.

Function declarations and MainQuiz Linked List structure

```c
// MainQuiz Module

// Linked List structure for storing Main Quiz questions, options and result
struct MainQuiz {
    char concept[20];
    char ques[250];
    char optionA[50];
    char optionB[50];
    char optionC[50];
    char optionD[50];
    char result[50];
    struct MainQuiz *NEXT;
};
struct MainQuiz *HEADMQ = NULL;

// function declarations
int takeMainQuiz(int);
void loadMainQuizFromFile();
void insertMainQuiz(char[], char[], char[], char[], char[], char[], char[]);
int displayMainQuizQuestions(char[], int);
void writeScoreToUserFile(char[], int);
```

Function: int takeMainQuiz(int totalScore)

```c
// function for choosing concept in which to take the Main Quiz
int takeMainQuiz(int totalScore) {
    int choice, result, testTaken=0;

    // load quiz questions from database
    loadMainQuizFromFile();

    system("cls");

    SetColorForText("\n\n\n\t\t\t\t\t\t\t     Choose A Topic To Take A Quiz On! \n", 2);

    printf("\n\n\t\t\t\t\t\t%c", 218);
    for (int i = 0; i < 60; i++) {
        printf("%c",196);
    }
    printf("%c\n", 191);
    printf("\t\t\t\t\t\t%c%c                                                %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                    1. C                        %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                                                %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                    2. PYTHON                   %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                                                %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                    3. OOPs                     %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                                                %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                    4. GO BACK                  %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c%c                                                %c%c\n", 179 ,179, 179, 179);
    printf("\t\t\t\t\t\t%c", 192);

    for (int i = 0; i < 60; i++) {
        printf("%c",196);
    }
```

```c
    printf("%c\n",217);

    SetColorForText("\n\tEnter Your Choice: ", 2);

    scanf("%d", &choice);

    system("cls");

    // based on choice chosen, we display the questions and store the result
    switch (choice) {
        case 1:;
            result = displayMainQuizQuestions("C", totalScore);
            testTaken = 1;
            break;
        case 2:;
            result = displayMainQuizQuestions("Python", totalScore);
            testTaken = 1;
            break;
```

```c
        case 3:;
            result = displayMainQuizQuestions("OOPS", totalScore);
            testTaken = 1;
            break;
        case 4:
            break;
        default:
            SetColorForText("\n\tINVALID ENTRY!", 4);
            delayTime(500);
            break;
    }

    if (testTaken){
        printf("\t\t\t\t\t\t\t\tYou Scored: %d Points!\n", (result-totalScore));
        SetColorForText("\n\t\t\t\t\t\t\t\tThank You For Taking The Quiz :) \n\t\t\t\t\t\t\t\tRedirecting You Back To The Main Menu!\n\n",
        delayTime(2500);
    }

    // result contains the point scored by the user
    return result;
}
```

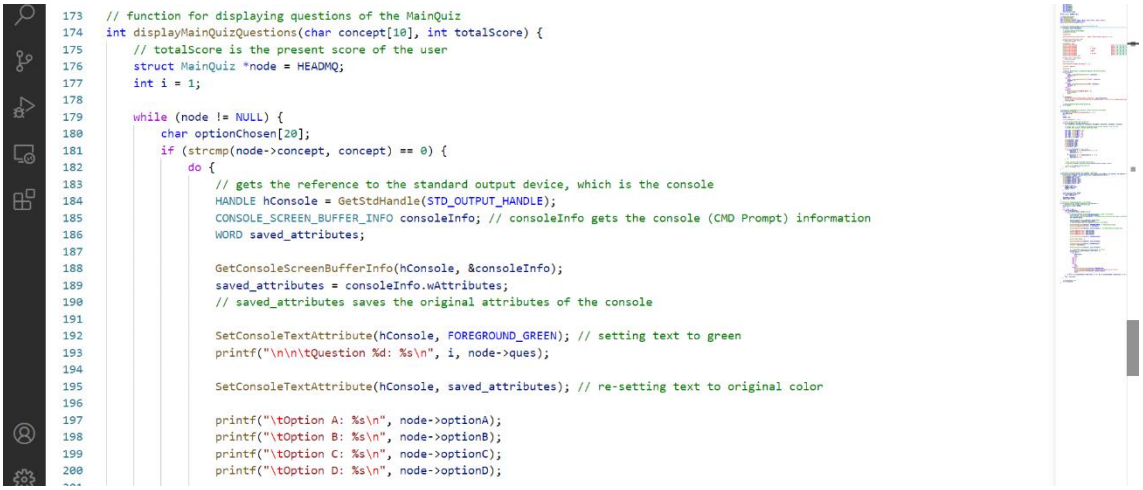
Function: void loadMainQuizFromFile()

```c
// function for loading main quiz questions, options and result from database
void loadMainQuizFromFile() {
    FILE *fptr = fopen("Data Files/mainQuizData.txt", "r");
    char buffer[500];
    int i;

    HEADMQ = NULL;

    printf("Loading Data.....\n");

    // First, we read line by line from the file
    while (fgets(buffer, 700, fptr) != NULL) {
        char concepts[20], question[250], optionA[50], optionB[50], optionC[50], optionD[50], result[50];

        // we have used "|" as the delimiter and based on that we split the data in the line into
        // concept name, question, the four options and result

        char *conc = strtok(buffer, "|");
        char *ques = strtok(NULL, "|");
        char *optA = strtok(NULL, "|");
        char *optB = strtok(NULL, "|");
        char *optC = strtok(NULL, "|");
        char *optD = strtok(NULL, "|");
        char *res = strtok(NULL, "|");

        strcpy(concepts, conc);
        strcpy(question, ques);
        strcpy(optionA, optA);
        strcpy(optionB, optB);
        strcpy(optionC, optC);
        strcpy(optionD, optD);
        strcpy(result, res);

        for (i = 0; question[i] != '\0'; i++) {
            if (question[i] == '\\' && question[i+1] == 't') {
                question[i] = ' ';
                question[i+1] = '\t';
            }
            if (question[i] == '\\' && question[i+1] == 'n') {
                question[i] = ' ';
                question[i+1] = '\n';
            }
        }

        // Then, we insert into the linked list structure
        insertMainQuiz(concepts, question, optionA, optionB, optionC, optionD, result);

        // This is for reading from the next line
        conc = strtok(NULL, "|");
    }
}
```

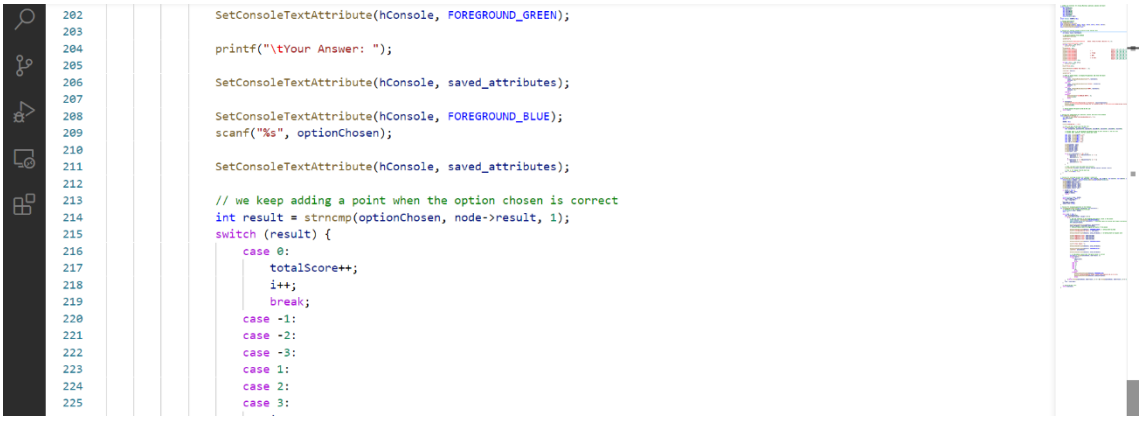
Function: void insertMainQuiz(char *Concept, char *Question, char *OptA, char *OptB, char *OptC, char *OptD, char *Res)

```c
// function for inserting new data into 'MainQuiz' linked list
void insertMainQuiz(char conc[20], char ques[250], char optA[20], char optB[20], char optC[20], char optD[20], char res[20]) {
    struct MainQuiz *newData = (struct MainQuiz*)malloc(sizeof(struct MainQuiz));
    strcpy(newData->concept, conc);
    strcpy(newData->ques, ques);
    strcpy(newData->optionA, optA);
    strcpy(newData->optionB, optB);
    strcpy(newData->optionC, optC);
    strcpy(newData->optionD, optD);
    strcpy(newData->result, res);

    if (HEADMQ == NULL) {
        newData->NEXT = NULL;
        HEADMQ = newData;
        return;
    }

    struct MainQuiz *temp = HEADMQ;
    while (temp->NEXT != NULL) {
        temp = temp->NEXT;
    }
    temp->NEXT = newData;
    newData->NEXT = NULL;
}
```

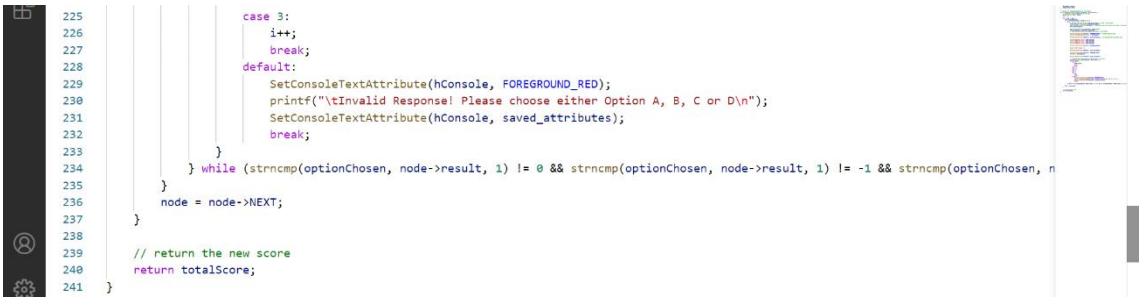
Function: int displayMainQuizQuestions(char *Concept, int totalScore)



```
// function for displaying questions of the MainQuiz
int displayMainQuizQuestions(char concept[10], int totalScore) {
    // totalScore is the present score of the user
    struct MainQuiz *node = HEADMQ;
    int i = 1;

    while (node != NULL) {
        char optionChosen[20];
        if (strcmp(node->concept, concept) == 0) {
            do {
                // gets the reference to the standard output device, which is the console
                HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                CONSOLE_SCREEN_BUFFER_INFO consoleInfo; // consoleInfo gets the console (CMD Prompt) information
                WORD saved_attributes;

                GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
                saved_attributes = consoleInfo.wAttributes;
                // saved_attributes saves the original attributes of the console

                SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN); // setting text to green
                printf("\n\n\tQuestion %d: %s\n", i, node->ques);

                SetConsoleTextAttribute(hConsole, saved_attributes); // re-setting text to original color

                printf("\tOption A: %s\n", node->optionA);
                printf("\tOption B: %s\n", node->optionB);
                printf("\tOption C: %s\n", node->optionC);
                printf("\tOption D: %s\n", node->optionD);
```



```
                SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);

                printf("\tYour Answer: ");

                SetConsoleTextAttribute(hConsole, saved_attributes);

                SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
                scanf("%s", optionChosen);

                SetConsoleTextAttribute(hConsole, saved_attributes);

                // we keep adding a point when the option chosen is correct
                int result = strncmp(optionChosen, node->result, 1);
                switch (result) {
                    case 0:
                        totalScore++;
                        i++;
                        break;
                    case -1:
                    case -2:
                    case -3:
                    case 1:
                    case 2:
                    case 3:
```



```
                    case 3:
                        i++;
                        break;
                    default:
                        SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
                        printf("\tInvalid Response! Please choose either Option A, B, C or D\n");
                        SetConsoleTextAttribute(hConsole, saved_attributes);
                        break;
                }
            } while (strncmp(optionChosen, node->result, 1) != 0 && strncmp(optionChosen, node->result, 1) != -1 && strncmp(optionChosen, n
        }
        node = node->NEXT;
    }

    // return the new score
    return totalScore;
}
```

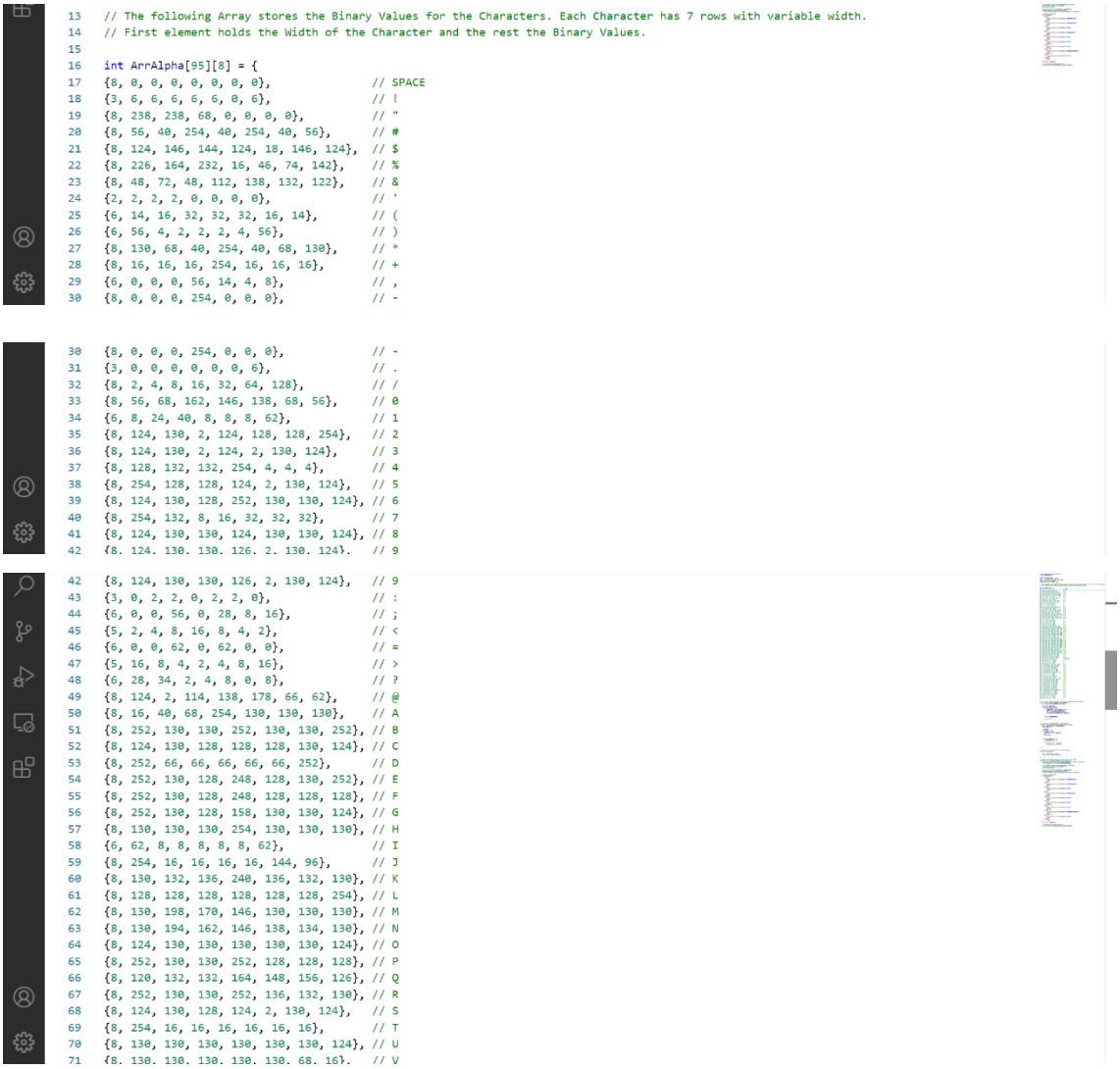
3.2.1.5. BANNER

The Banner module was created for stylistically printing our project's name when we run our console application.

Function declarations and definitions

```
// Banner Module
// For displaying the headings in a stylish manner

// defining printing speed of banner
#define BANNERSPEED 50

// function declaration
void    getBanner(char *, char);
void    DisplayBannerRow(int, int, char);
void delayTime(int seconds);
void SetColorForText(char*, int);

```

Global variables



```
// The following Array stores the Binary Values for the Characters. Each Character has 7 rows with variable width.
// First element holds the Width of the Character and the rest the Binary Values.

int ArrAlpha[95][8] = {
{8, 0, 0, 0, 0, 0, 0, 0},              // SPACE
{3, 6, 6, 6, 6, 6, 0, 6},              // !
{8, 238, 238, 68, 0, 0, 0, 0},         // "
{8, 56, 40, 254, 40, 254, 40, 56},     // #
{8, 124, 146, 144, 124, 18, 146, 124}, // $
{8, 226, 164, 232, 16, 46, 74, 142},   // %
{8, 48, 72, 48, 112, 138, 132, 122},   // &
{2, 2, 2, 2, 0, 0, 0, 0},              // '
{6, 14, 16, 32, 32, 32, 16, 14},       // (
{6, 56, 4, 2, 2, 2, 4, 56},            // )
{8, 130, 68, 40, 254, 40, 68, 130},    // *
{8, 16, 16, 16, 254, 16, 16, 16},      // +
{6, 0, 0, 0, 56, 14, 4, 8},            // ,
{8, 0, 0, 0, 254, 0, 0, 0},            // -
{8, 0, 0, 0, 254, 0, 0, 0},            // -
{3, 0, 0, 0, 0, 0, 0, 6},              // .
{8, 2, 4, 8, 16, 32, 64, 128},         // /
{8, 56, 68, 162, 146, 138, 68, 56},    // 0
{6, 8, 24, 40, 8, 8, 8, 62},           // 1
{8, 124, 130, 2, 124, 128, 128, 254},  // 2
{8, 124, 130, 2, 124, 2, 130, 124},    // 3
{8, 128, 132, 132, 254, 4, 4, 4},      // 4
{8, 254, 128, 128, 124, 2, 130, 124},  // 5
{8, 124, 130, 128, 252, 130, 130, 124}, // 6
{8, 254, 132, 8, 16, 32, 32, 32},      // 7
{8, 124, 130, 130, 124, 130, 130, 124}, // 8
{8, 124, 130, 130, 126, 2, 130, 124},  // 9
{3, 0, 2, 2, 0, 2, 2, 0},              // :
{6, 0, 0, 56, 0, 28, 8, 16},           // ;
{5, 2, 4, 8, 16, 8, 4, 2},             // <
{6, 0, 0, 62, 0, 62, 0, 0},            // =
{5, 16, 8, 4, 2, 4, 8, 16},            // >
{6, 28, 34, 2, 4, 8, 0, 8},            // ?
{8, 124, 2, 114, 138, 178, 66, 62},    // @
{8, 16, 40, 68, 254, 130, 130, 130},   // A
{8, 252, 130, 130, 252, 130, 130, 252}, // B
{8, 124, 130, 128, 128, 128, 130, 124}, // C
{8, 252, 66, 66, 66, 66, 66, 252},     // D
{8, 252, 130, 128, 248, 128, 130, 252}, // E
{8, 252, 130, 128, 248, 128, 128, 128}, // F
{8, 252, 130, 128, 158, 130, 130, 124}, // G
{8, 130, 130, 130, 254, 130, 130, 130}, // H
{6, 62, 8, 8, 8, 8, 8, 62},            // I
{8, 254, 16, 16, 16, 16, 144, 96},     // J
{8, 130, 132, 136, 240, 136, 132, 130}, // K
{8, 128, 128, 128, 128, 128, 128, 254}, // L
{8, 130, 198, 170, 146, 130, 130, 130}, // M
{8, 130, 194, 162, 146, 138, 134, 130}, // N
{8, 124, 130, 130, 130, 130, 130, 124}, // O
{8, 252, 130, 130, 252, 128, 128, 128}, // P
{8, 120, 132, 132, 164, 148, 156, 126}, // Q
{8, 252, 130, 130, 252, 136, 132, 130}, // R
{8, 124, 130, 128, 124, 2, 130, 124},  // S
{8, 254, 16, 16, 16, 16, 16, 16},      // T
{8, 130, 130, 130, 130, 130, 130, 124}, // U
{8, 130, 130, 130, 130, 130, 68, 16},  // V
```

```
    {8, 130, 130, 130, 130, 130, 68, 16},   // V
    {8, 130, 130, 146, 146, 146, 146, 108}, // W
    {8, 130, 68, 40, 16, 40, 68, 130},      // X
    {8, 130, 68, 40, 16, 16, 16, 16},       // Y
    {8, 254, 132, 8, 16, 32, 66, 254},      // Z
    {6, 30, 32, 32, 32, 32, 32, 30},        // [
    {8, 128, 64, 32, 16, 8, 4, 2},          // BACKSLASH
    {8, 60, 2, 2, 2, 2, 2, 60},             // ]
    {7, 24, 36, 66, 0, 0, 0, 0},            // ^
    {7, 0, 0, 0, 0, 0, 0, 126},             // _
    {4, 14, 6, 2, 0, 0, 0, 0},              // `
    {8, 0, 120, 132, 4, 124, 132, 126},     // a
    {7, 0, 64, 64, 64, 124, 66, 124},       // b
    {7, 0, 60, 66, 64, 64, 66, 60},         // c
    {7, 0, 2, 2, 62, 66, 66, 60},           // d
    {7, 0, 60, 66, 66, 124, 64, 62},        // e
    {7, 0, 14, 16, 124, 16, 16, 16},        // f
    {7, 0, 60, 66, 66, 62, 2, 126},         // g
    {8, 0, 128, 128, 184, 196, 130, 130},   // h
    {4, 0, 4, 0, 4, 4, 4, 14},              // i
    {5, 0, 2, 0, 2, 2, 2, 28},              // j
    {5, 0, 16, 18, 20, 30, 18, 18},         // k
    {4, 0, 12, 4, 4, 4, 4, 14},             // l
    {8, 0, 124, 146, 146, 146, 130, 130},   // m
    {7, 0, 124, 34, 34, 34, 34, 34},        // n
    {7, 0, 24, 36, 66, 66, 36, 24},         // o
    {7, 0, 60, 66, 66, 124, 64, 64},        // p
    {7, 0, 48, 72, 72, 56, 10, 12},         // q
    {7, 0, 92, 34, 32, 32, 32, 32},         // r
    {7, 0, 60, 66, 60, 2, 66, 60},          // s
    {5, 0, 8, 28, 8, 8, 10, 12},            // t
    {8, 0, 66, 66, 66, 66, 66, 60},         // u
    {6, 0, 34, 34, 34, 34, 20, 8},          // v
    {8, 0, 130, 130, 146, 146, 170, 68},    // w
    {7, 0, 66, 36, 24, 24, 36, 66},         // x
    {6, 0, 34, 34, 30, 2, 2, 60},           // y
    {7, 0, 126, 4, 8, 16, 32, 126},         // z
    {5, 14, 8, 8, 24, 8, 8, 14},            // {
    {2, 2, 2, 2, 0, 2, 2, 2},               // |
    {5, 28, 4, 4, 6, 4, 4, 28},             // }
    {7, 96, 82, 12, 0, 0, 0, 0}             // ~
};
```

Function: void getBanner(char *ToBePrintedStr, char DisplayChar)

```
// function for printing the banner with given string and display character
void    getBanner(char  *AlphaStr, char DispChar) {
    int i, Len, Row, AlphaIndex, Width, Num;

    Len = strlen(AlphaStr);
    for (Row=0; Row<7; Row++) {
        for (i=0; i<Len; i++) {
            AlphaIndex = (int) AlphaStr[i] - 32;
            Width = ArrAlpha[AlphaIndex][0];
            Num = ArrAlpha[AlphaIndex][Row+1];
            DisplayBannerRow(Width, Num, DispChar);
        }

        delayTime(BANNERSPEED);

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

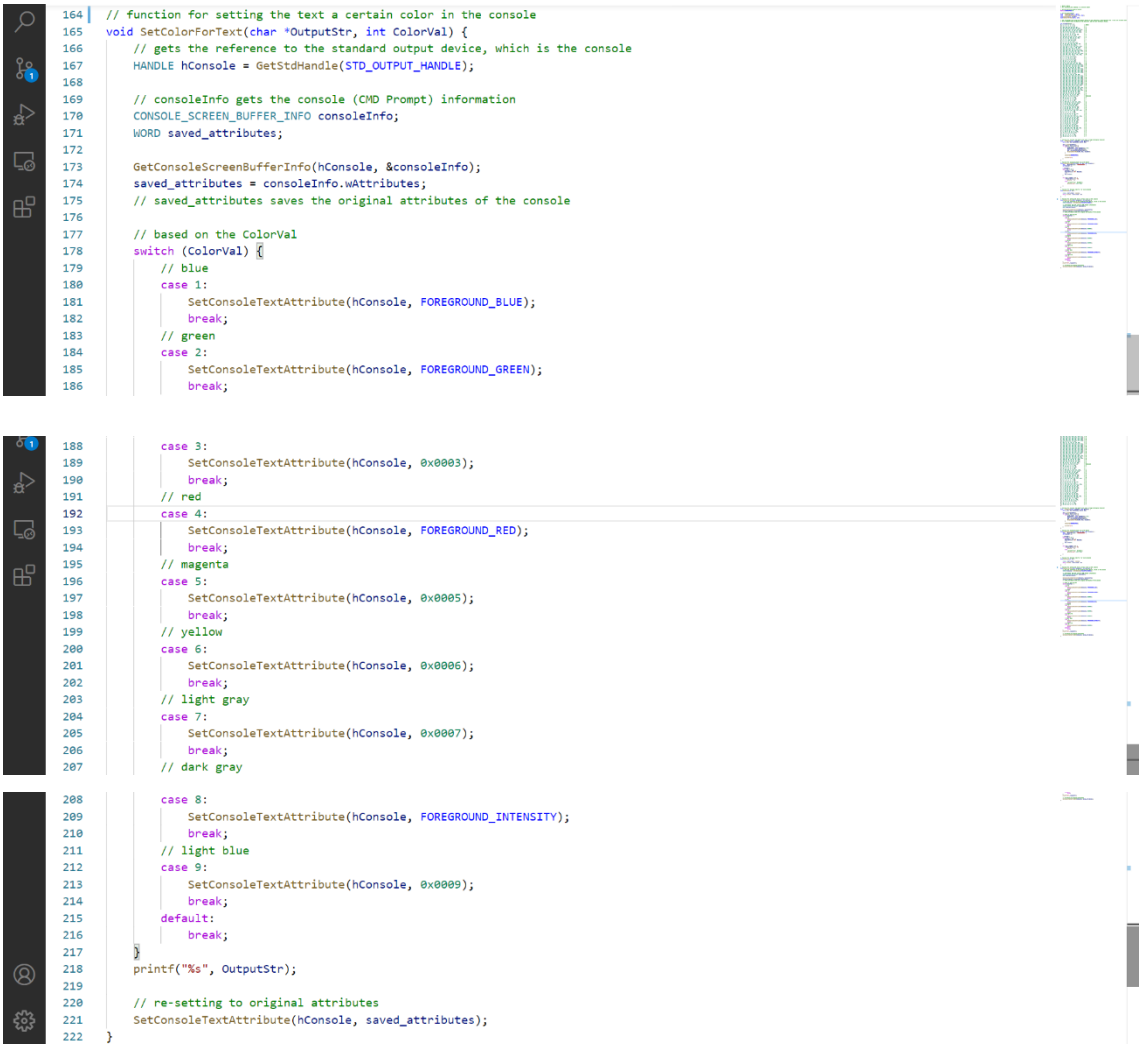
Function: void DisplayBannerRow(int Width, int Num, char DisplayChar)

```
// function for displaying each row in the banner
void DisplayBannerRow(int Width, int Num, char DispChar) {
    char    BinaryStr[12] = "000000000000" ;
    int Reminder, i;

    i = Width-1;
    while (Num != 0) {
        Reminder = Num % 2;
        BinaryStr[i] = '0'+ Reminder;
        i--;
        Num = Num/2;
    }

    for (i=0; i<Width; i++) {
        if(BinaryStr[i] == '0')
            printf("  ");
        else
            //printf("|%c", DispChar);
            printf("%c%c", 179, 179);

    }
}
```

Function: void delayTime(int milli_seconds)

```
// function for delaying time for 'ms' milli-seconds
void delayTime(int ms) {

    clock_t start_time = clock();
    while (clock() < start_time + ms)
        ;
}
```

Function: void SetColorForText(char *ToBePrintedStr, int ColorVal)



```
// function for setting the text a certain color in the console
void SetColorForText(char *OutputStr, int ColorVal) {
    // gets the reference to the standard output device, which is the console
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

    // consoleInfo gets the console (CMD Prompt) information
    CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
    WORD saved_attributes;

    GetConsoleScreenBufferInfo(hConsole, &consoleInfo);
    saved_attributes = consoleInfo.wAttributes;
    // saved_attributes saves the original attributes of the console

    // based on the ColorVal
    switch (ColorVal) {
        // blue
        case 1:
            SetConsoleTextAttribute(hConsole, FOREGROUND_BLUE);
            break;
        // green
        case 2:
            SetConsoleTextAttribute(hConsole, FOREGROUND_GREEN);
            break;
```



```
        case 3:
            SetConsoleTextAttribute(hConsole, 0x0003);
            break;
        // red
        case 4:
            SetConsoleTextAttribute(hConsole, FOREGROUND_RED);
            break;
        // magenta
        case 5:
            SetConsoleTextAttribute(hConsole, 0x0005);
            break;
        // yellow
        case 6:
            SetConsoleTextAttribute(hConsole, 0x0006);
            break;
        // light gray
        case 7:
            SetConsoleTextAttribute(hConsole, 0x0007);
            break;
        // dark gray
```

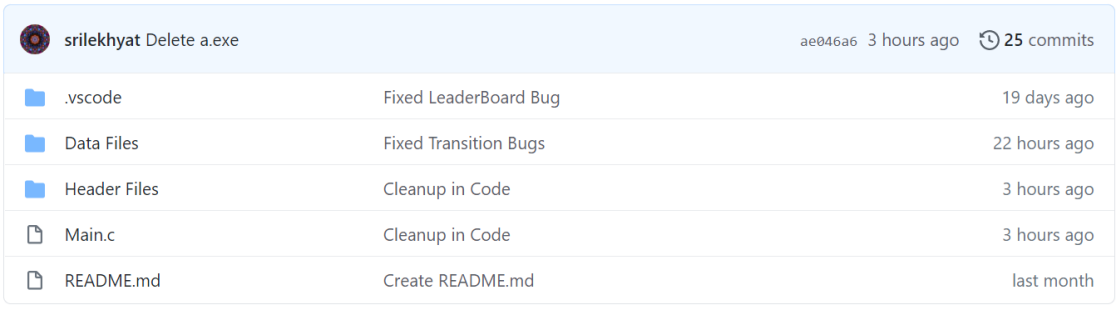


```
        case 8:
            SetConsoleTextAttribute(hConsole, FOREGROUND_INTENSITY);
            break;
        // light blue
        case 9:
            SetConsoleTextAttribute(hConsole, 0x0009);
            break;
        default:
            break;
    }
    printf("%s", OutputStr);

    // re-setting to original attributes
    SetConsoleTextAttribute(hConsole, saved_attributes);
}
```

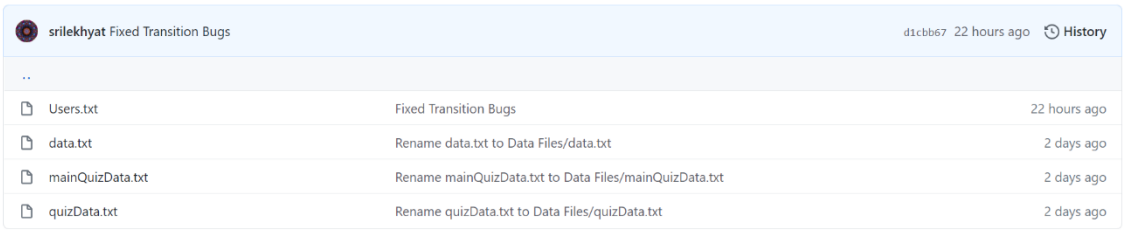
3.2.2. GITHUB/FOLDER STRUCTURE

We have segregated the files we have used based on the usage. We made 2 folders, one will comprise all the Header Files and the other will consist of all the Data files. We also have a README which has a brief description of our project. The Main C file is not part of any folder.

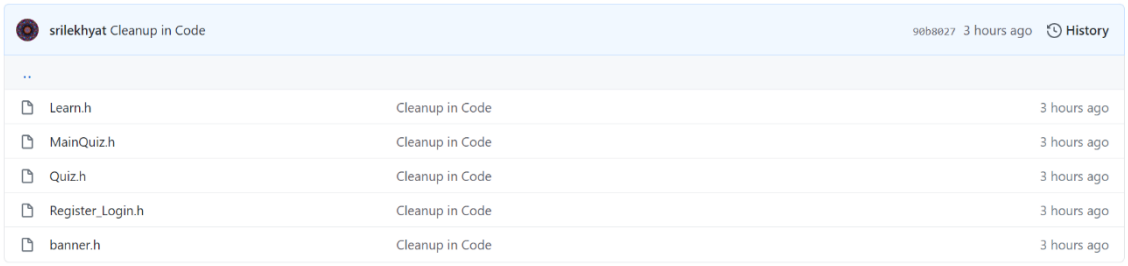
GitHub Repo Link: https://github.com/srilekhyat/Codiac-MiniProject



Within the Data Files, we have:



Within the Header Files, we have:



3.3. TESTING

Testing is a method to check whether the actual product matches the expected requirements and to ensure that the product is defect-free. This process involves execution of various parts of the product either using manual or automated tools. The purpose is to identify errors, gaps or missing requirements in contrast to the actual requirements.

3.3.1. TEST PLAN

We approached testing our console application by analysing each module separately. First, we coded the requirements and then manually tested each feature present in the module to cover any gaps that might occur.

3.3.2. USER TEST CASES

The user has 5 major functionalities: Register, Login, Learn a Concept, Take a Quiz and View the Leader board. Below are the test cases which we have compiled together manually.

3.3.2.1. REGISTER

Test Case ID: TC01		**Use Case ID:** UC01
Test Case Title: User – Register		
Test Case Description: User attempts to register into CODIAC with a username which already exists.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for user to enter username 2. User enters a username which already exists.	System should display "Invalid credentials".	The System displays "Username already exists, please choose another one".

Test Case ID: TC02	**Use Case ID:**
Test Case Title: User – Register	

Test Case Description: User attempts to register into CODIAC with improper password.		UC01
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for user to enter password. 2. User enters a password which does meet the criteria.	System should display "Invalid credentials" with respective valid errors.	System displays "Invalid Credentials" along with the error to fix

Test Case ID: TC03		**Use Case ID:** UC01
Test Case Title: User – Register		
Test Case Description: While registering, the user does not re-enter the correct and same password.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for user to re-enter password. 2. User enters a password which does match the previously entered password.	System should display an error message.	The System displays "Password does not match".

3.3.2.2. LOGIN

Test Case ID: TC04		**Use Case ID:** UC02
Test Case Title: User - Login		
Test Case Description: User attempts to login into CODIAC without registering or with invalid username		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for user to login 2. User enters a username and password which is not present in the database.	System should display a message referring to users not present in the database.	System displays "User Not Found!".

Test Case ID: TC05	**Use Case ID:**
Test Case Title: User – Login	

Test Case Description: User attempts to login into CODIAC with invalid password		UC02
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for user to login 2. User enters invalid password.	System should display "Invalid credentials".	System displays "Invalid Credentials".

3.3.2.3. LEARN A CONCEPT

Test Case ID: TC06		**Use Case ID:** UC03
Test Case Title: User – Learn a Concept		
Test Case Description: While choosing the learn a concept choice, the user chooses an invalid option.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays a prompt for the user to choose an option. 2. User enters invalid choice.	System should display "Invalid Entry!".	System displays "Invalid Entry!".

Test Case ID: TC07		**Use Case ID:** UC03
Test Case Title: User - Learn a Concept		
Test Case Description: While choosing a concept, the user chooses an invalid option.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays a list of concepts to choose from. 2. User chooses an option which is not present.	System should error display message.	System displays "Invalid Entry" message.

Test Case ID: TC08		**Use Case ID:** UC03
Test Case Title: User - Learn a Concept		
Test Case Description: While choosing a topic within the concept, the user chooses an invalid option.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays a list of topics to learn from after choosing a concept.	System should error display message.	System displays "Invalid Entry" message.

2. User chooses an option which is not present.		

Test Case ID: TC09		**Use Case ID:** UC03
Test Case Title: User - Learn a Concept		
Test Case Description: After choosing a topic, display the topic of choice.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses a topic from the list provided.	Information regarding the topic should be displayed	Information regarding the topic is displayed.

Test Case ID: TC10		**Use Case ID:** UC03
Test Case Title: User - Learn a Concept		
Test Case Description: After reading a topic, the user chooses to take a Mini-Assessment based on the information read.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses to take a Mini-Assessment. 2. System displays the relevant questions of the topic chosen.	Relevant questions should be displayed.	Relevant questions are displayed along with a space to answer.

Test Case ID: TC11		**Use Case ID:** UC03
Test Case Title: User - Learn a Concept		
Test Case Description: After reading a topic, the user chooses to take a Mini-Assessment based on the information read. User enters invalid choices in the Mini-Assessment		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses to take a Mini-Assessment. 2. System displays the relevant questions of the topic chosen. 3. User chooses invalid options (options other than A, B, C or D).	Questions should be displayed and after taking input from the user, relevant error messages should be displayed.	After taking input from the user, a relevant error message is displayed. The question is displayed again to re-answer.

3.3.2.4. TAKE A QUIZ

Test Case ID: TC12		**Use Case ID:** UC04
Test Case Title: User - Take a Quiz		
Test Case Description: User chooses to take the Main Quiz and is requested to choose an option. User chooses an invalid option.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses to take the Main Quiz. 2. System displays the concepts to take a quiz from. 3. User chooses an invalid option.	System should display an error message.	System displays an error message – "Invalid Entry"

Test Case ID: TC13		**Use Case ID:** UC04
Test Case Title: User - Take a Quiz		
Test Case Description: User chooses to take the Main Quiz and while taking the quiz, the user chooses an invalid option.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses a concept to take the quiz. 2. While taking the quiz, the user enters an invalid option (Any option other than A, B, C or D).	After taking the input from the user, the system should display an error message.	After taking input from the user, a relevant error message is displayed. The question is displayed again to re-answer.

3.3.2.5. VIEW POINTS LEADERBOARD

Test Case ID: TC14		**Use Case ID:** UC05
Test Case Title: User - View Points Leader board		
Test Case Description: User chooses to view the points leader board		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses to view the leader board.	The leader board should be displayed.	The leader board is displayed.

Test Case ID: TC15		**Use Case ID:** UC05
Test Case Title: User - View Points Leader board		
Test Case Description: User chooses to view the points leader board but there aren't any users present.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. User chooses to view the leader board.	Error message is displayed.	Message is displayed referring to lack of users.

3.3.3. ADMIN TEST CASES

The Admin has 5 major functionalities: Login, View User Table, Edit Users Details, Delete User and View Points Leader board. Below are the test cases which we have compiled together manually.

3.3.3.1. LOGIN

Test Case ID: TC16		**Use Case ID:** UC06
Test Case Title: Admin - Login		
Test Case Description: Admin attempts to login into CODIAC with invalid username.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for Admin to login. 2. Admin enters invalid username.	System should display a message referring to invalid username.	System displays "User not found" error message.

Test Case ID: TC17		**Use Case ID:** UC06
Test Case Title: Admin - Login		
Test Case Description: Admin attempts to login into CODIAC with invalid password.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt for Admin to login. 2. Admin enters invalid password.	System should display a message referring to invalid credentials.	System displays "Invalid Credentials" as an error message.

3.3.3.2. VIEW USER TABLE

Test Case ID: TC18		**Use Case ID:** UC07
Test Case Title: Admin - View User Table		
Test Case Description: Admin attempts to view the User Table containing all information with no users present.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. Admin chooses to view the user table.	System should display a message referring to the lack of users.	System displays lack of users as an error message.

Test Case ID: TC19		**Use Case ID:** UC07
Test Case Title: Admin - View User Table		
Test Case Description: Admin attempts to view the User Table containing all information.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. Admin chooses to view the user table.	System should display the user table.	System displays the user table.

3.3.3.3. EDIT USERS DETAILS

Test Case ID: TC20		**Use Case ID:** UC08
Test Case Title: Admin – Edit Users Details		
Test Case Description: Admin attempts to edit details of a user who does not exist.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt to enter username of user whose details you want to edit. 2. User enters an invalid username.	System should display an error message regarding invalid username.	System displays "User Not Found!" as an error message.

Test Case ID: TC21		**Use Case ID:** UC08
Test Case Title: Admin – Edit Users Details		
Test Case Description: Admin attempts to edit details of a user who does exist.		
Test Steps:	**Expected Result:**	**Actual Result:**

1. System displays prompt to enter username of user whose details you want to edit. 2. User enters a valid username.	System should display choices to the Admin of what to change.	System displays three choices: Password, first name and last name for Admin to edit.

3.3.3.4. DELETE USER

Test Case ID: TC22		**Use Case ID:** UC09
Test Case Title: Admin – Delete User		
Test Case Description: Admin attempts to delete a user who does not exist.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt to enter username of user Admin wants to delete. 2. User enters an invalid username.	System should display an error message regarding invalid username.	System displays "User Not Found!" as an error message.

Test Case ID: TC23		**Use Case ID:** UC09
Test Case Title: Admin – Delete User		
Test Case Description: Admin attempts to delete a user who does exist.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. System displays prompt to enter username of user whose details you want to edit. 2. User enters a valid username.	System should display that the user has been deleted.	System displays that the user has been deleted successfully.

3.3.3.5. VIEW POINTS LEADER BOARD

Test Case ID: TC24		**Use Case ID:** UC10
Test Case Title: Admin - View Points Leader board		
Test Case Description: Admin chooses to view the points leader board		
Test Steps:	**Expected Result:**	**Actual Result:**

1. Admin chooses to view the leader board.	The leader board should be displayed.	The leader board is displayed.

Test Case ID: TC25		**Use Case ID:** UC10
Test Case Title: Admin - View Points Leader board		
Test Case Description: Admin chooses to view the points leader board but there aren't any users present.		
Test Steps:	**Expected Result:**	**Actual Result:**
1. Admin chooses to view the leader board.	Error message is displayed.	Message is displayed referring to lack of users.

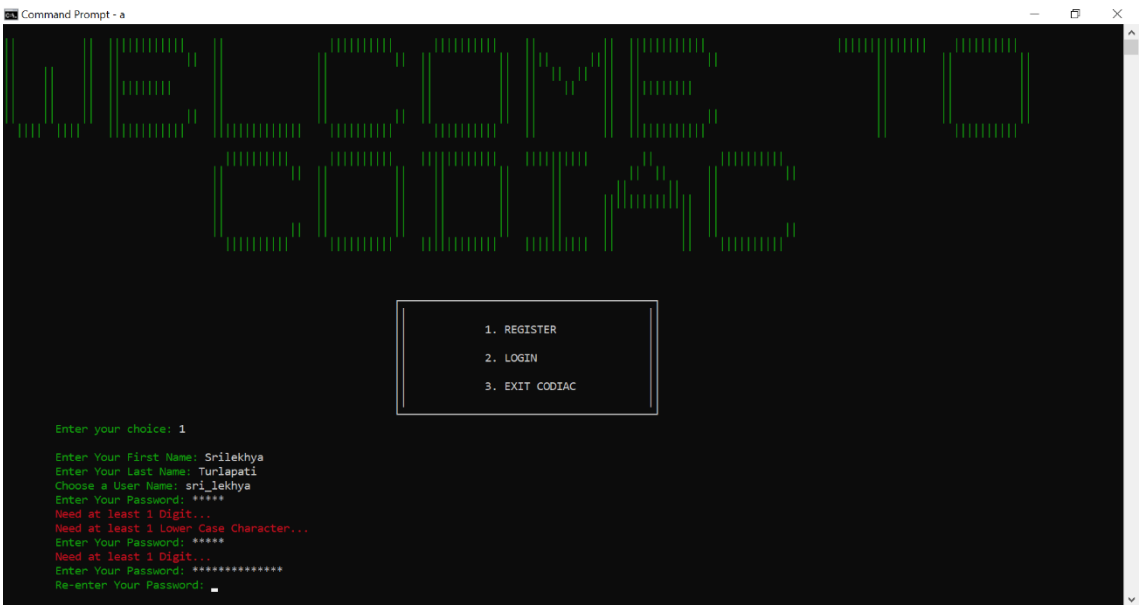
4. RESULTS

We have successfully developed a platform for users of all age groups to utilize and improve their skill-set. Below are the output screenshots of the test cases mentioned.

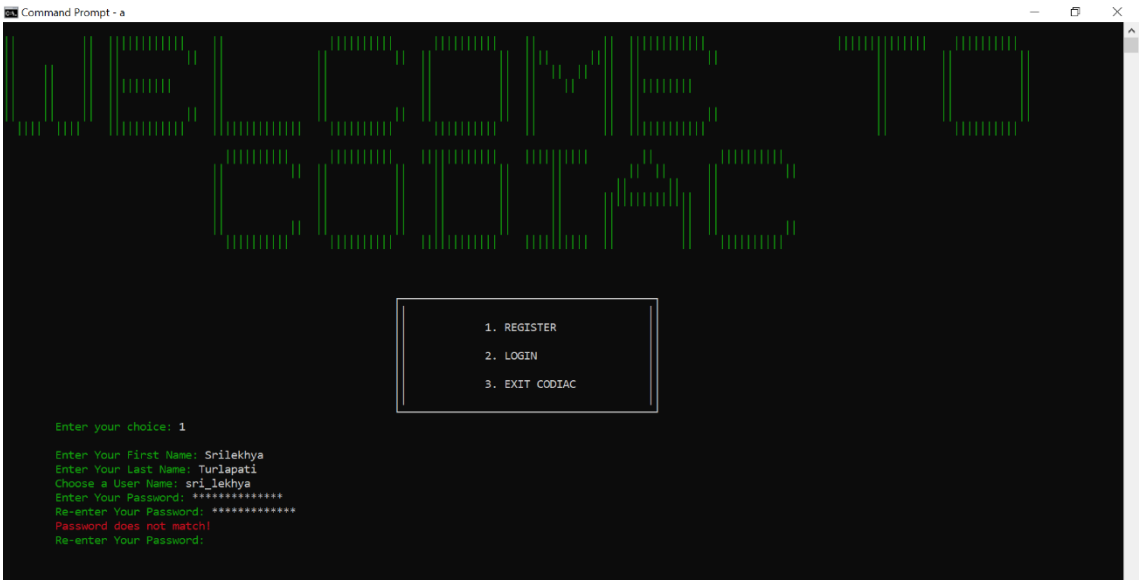
4.1. USER TEST CASE RESULTS

Test Case 1:

Test Case 2:

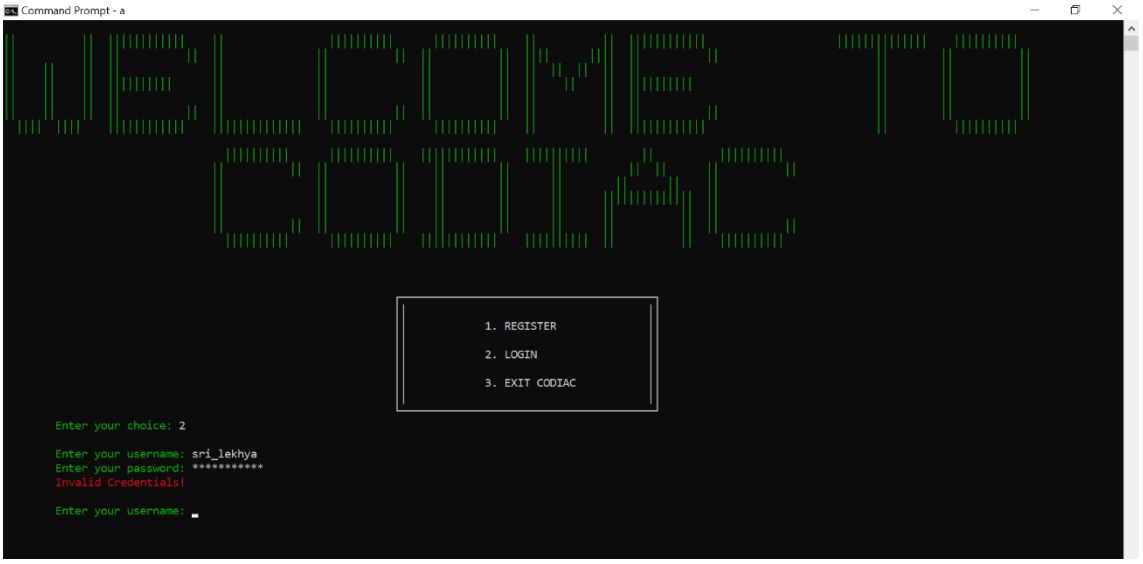


Test Case 3:



Test Case 4:

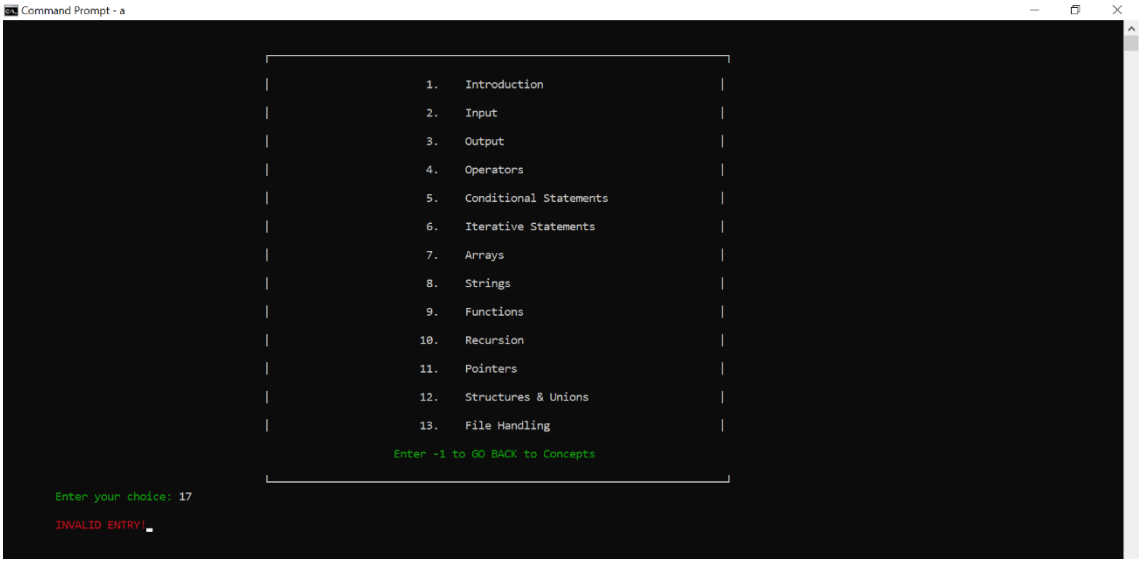
Test Case 5:



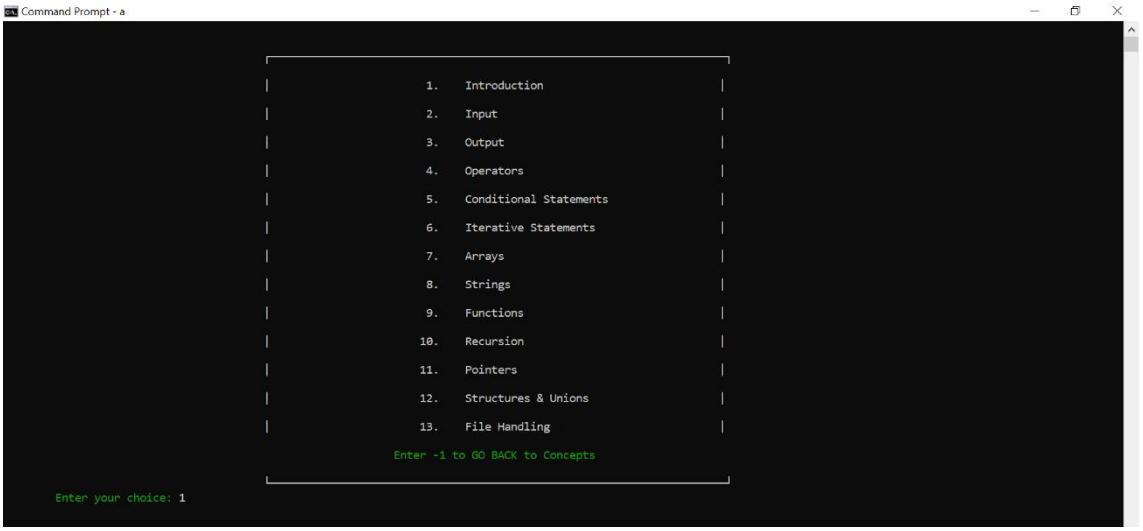
Test Case 6:

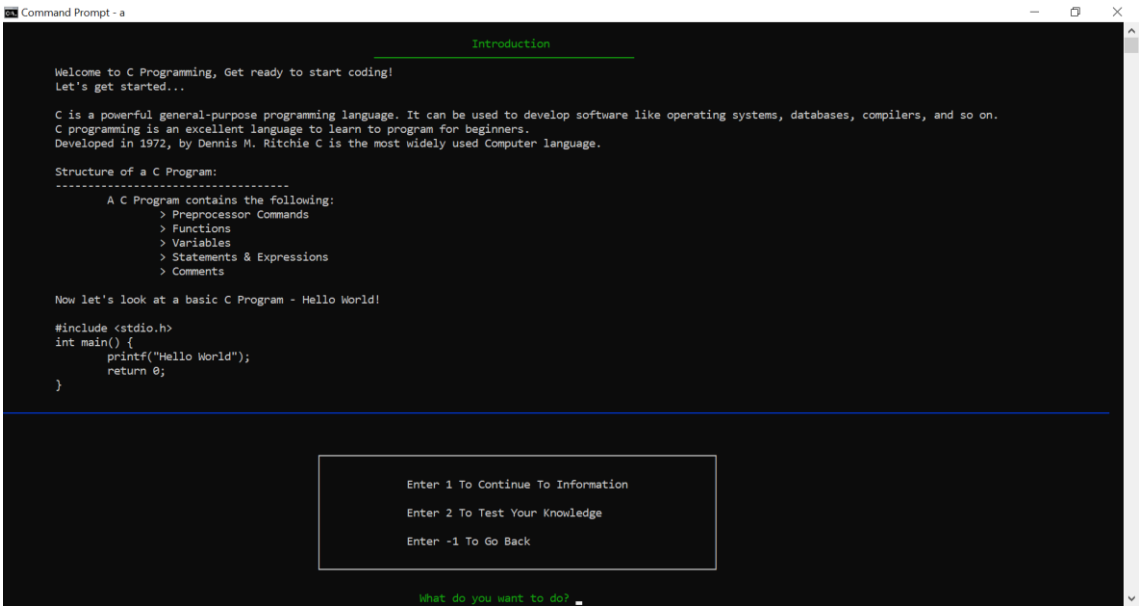
Test Case 7:

Test Case 8:



Test Case 9:




Command Prompt - a
Introduction
Welcome to C Programming, Get ready to start coding!
Let's get started...
C is a powerful general-purpose programming language. It can be used to develop software like operating systems, databases, compilers, and so on.
C programming is an excellent language to learn to program for beginners.
Developed in 1972, by Dennis M. Ritchie C is the most widely used Computer language.
Structure of a C Program:
A C Program contains the following:
> Preprocessor Commands
> Functions
> Variables
> Statements & Expressions
> Comments
Now let's look at a basic C Program - Hello World!
#include <stdio.h>
int main() {
printf("Hello World");
return 0;
}
Enter 1 To Continue To Information
Enter 2 To Test Your Knowledge
Enter -1 To Go Back
What do you want to do?

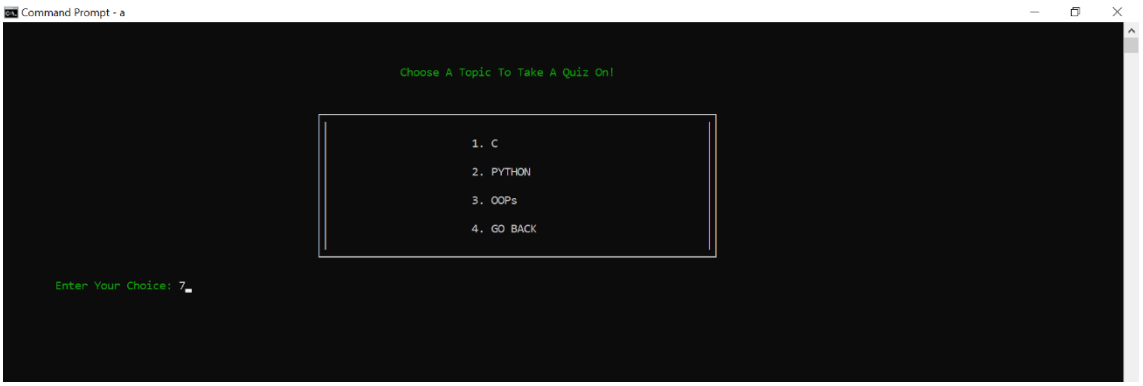
Test Case 10:


Command Prompt - a
Enter 1 To Continue To Information
Enter 2 To Test Your Knowledge
Enter -1 To Go Back
What do you want to do? 2


Command Prompt - a
Let's Test Your Knowledge!
Question 1: Who founded C Language?
Option A: John Backus
Option B: Kathleen Booth
Option C: Dennis Ritchie
Option D: Guido Van Rossum
Your Answer:

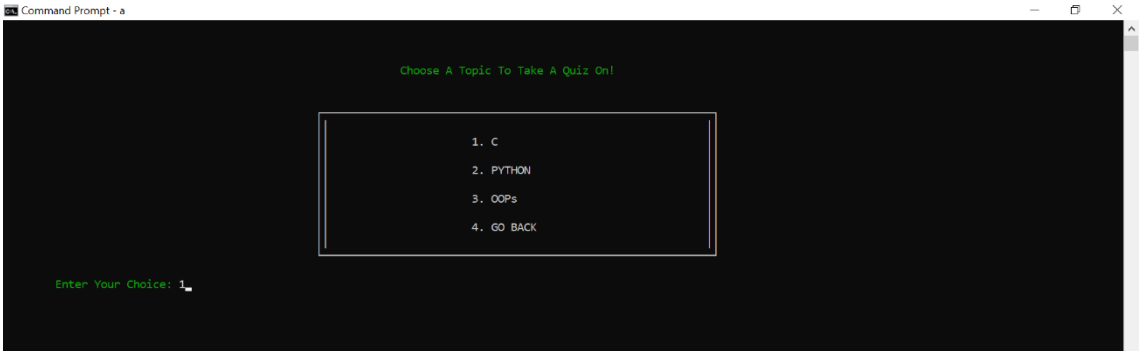
Test Case 11:

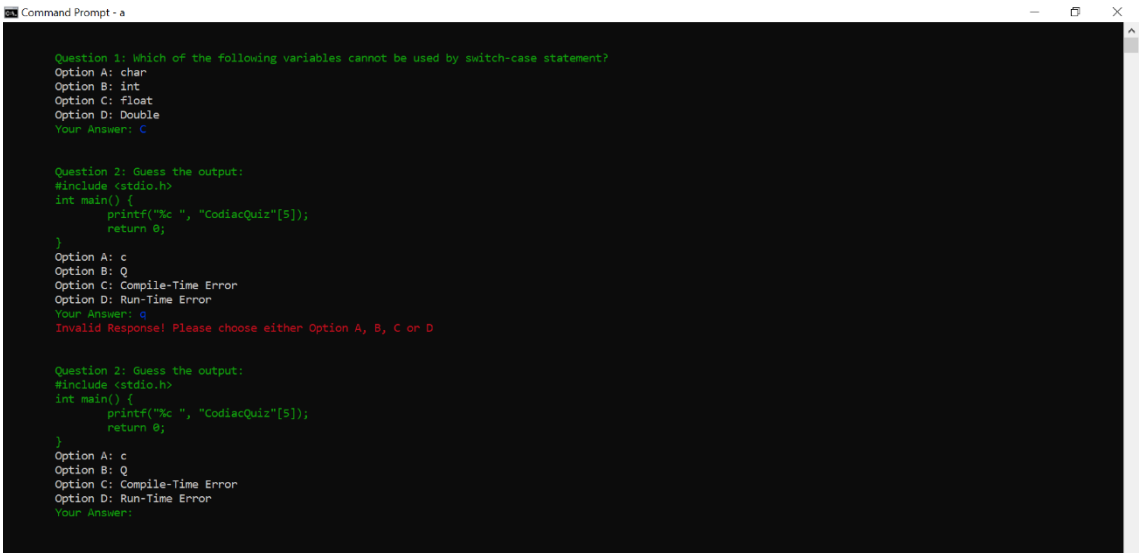
Test Case 12:

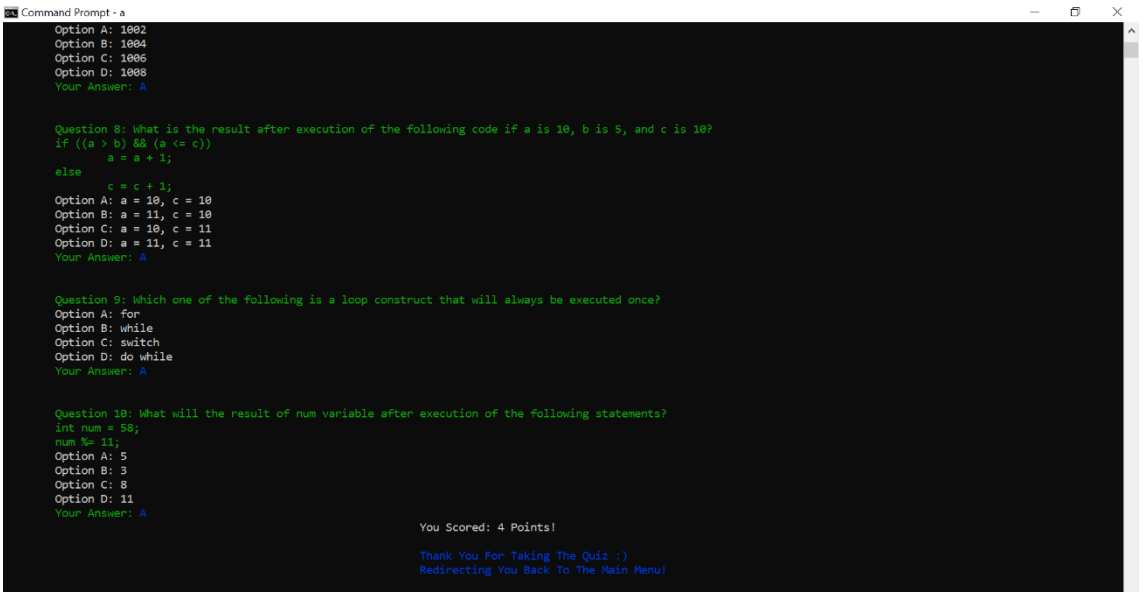



Command Prompt - a
INVALID ENTRY!

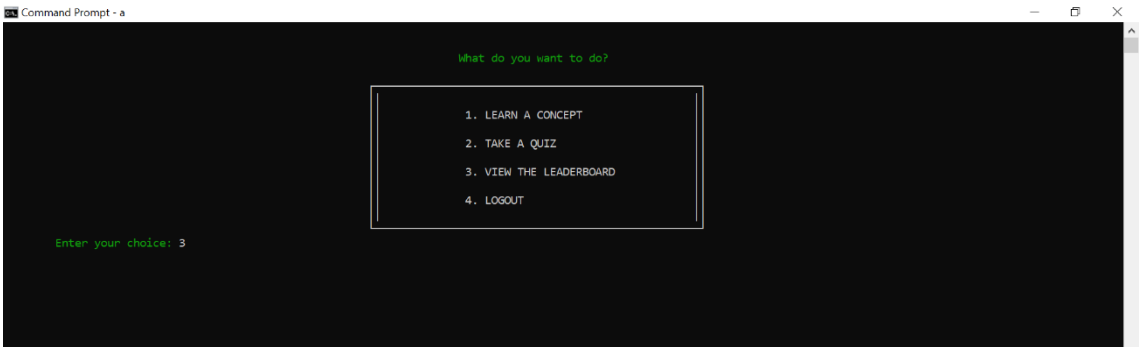
Test Case 13:

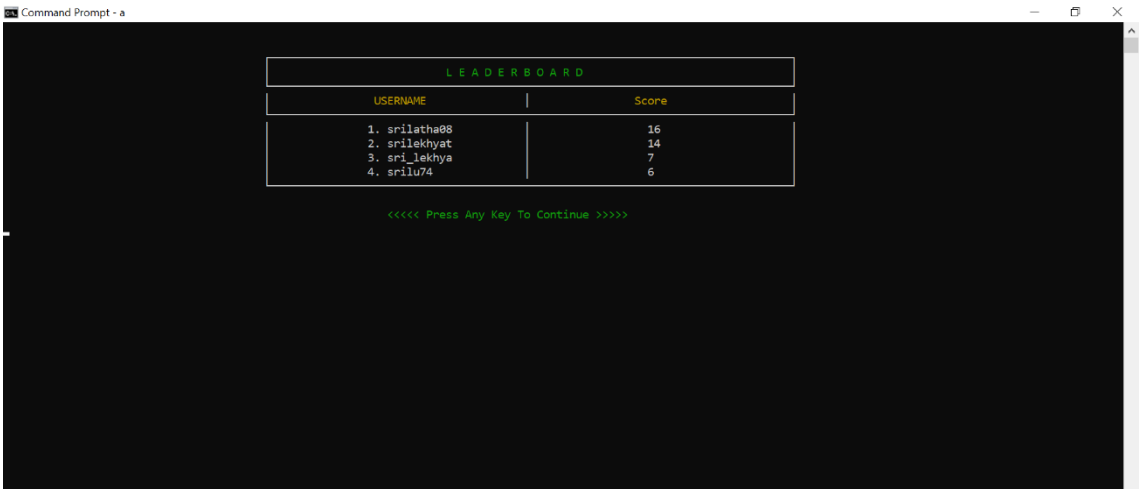

Command Prompt - a
Choose A Topic To Take A Quiz On!
1. C
2. PYTHON
3. OOPs
4. GO BACK
Enter Your Choice: 1


Command Prompt - a
Question 1: Which of the following variables cannot be used by switch-case statement?
Option A: char
Option B: int
Option C: float
Option D: Double
Your Answer: C
Question 2: Guess the output:
#include <stdio.h>
int main() {
printf("%c ", "CodiacQuiz"[5]);
return 0;
}
Option A: c
Option B: Q
Option C: Compile-Time Error
Option D: Run-Time Error
Your Answer: q
Invalid Response! Please choose either Option A, B, C or D
Question 2: Guess the output:
#include <stdio.h>
int main() {
printf("%c ", "CodiacQuiz"[5]);
return 0;
}
Option A: c
Option B: Q
Option C: Compile-Time Error
Option D: Run-Time Error
Your Answer:

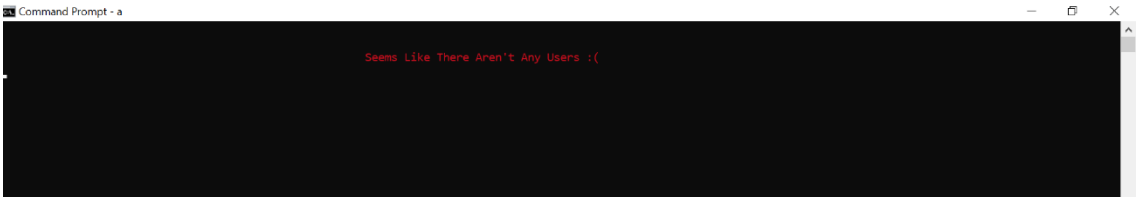

Command Prompt - a
Option A: 1002
Option B: 1004
Option C: 1006
Option D: 1008
Your Answer: A
Question 8: What is the result after execution of the following code if a is 10, b is 5, and c is 10?
if ((a > b) && (a <= c))
a = a + 1;
else
c = c + 1;
Option A: a = 10, c = 10
Option B: a = 11, c = 10
Option C: a = 10, c = 11
Option D: a = 11, c = 11
Your Answer: A
Question 9: Which one of the following is a loop construct that will always be executed once?
Option A: for
Option B: while
Option C: switch
Option D: do while
Your Answer: A
Question 10: What will the result of num variable after execution of the following statements?
int num = 58;
num %= 11;
Option A: 5
Option B: 3
Option C: 8
Option D: 11
Your Answer: A
You Scored: 4 Points!
Thank You For Taking The Quiz :)
Redirecting You Back To The Main Menu!

Test Case 14:


Command Prompt - a
What do you want to do?
1. LEARN A CONCEPT
2. TAKE A QUIZ
3. VIEW THE LEADERBOARD
4. LOGOUT
Enter your choice: 3

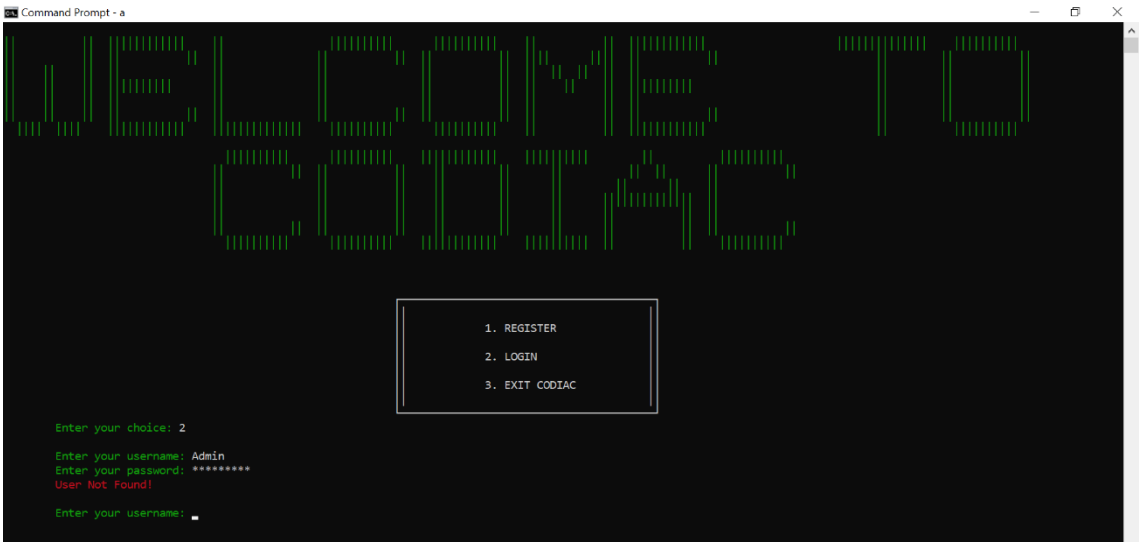

Command Prompt - a
L E A D E R B O A R D
USERNAME
Score
1. srilatha08
2. srilekhyat
3. sri_lekhya
4. srilu74
16
14
7
6
<<<<< Press Any Key To Continue >>>>>

Test Case 15:

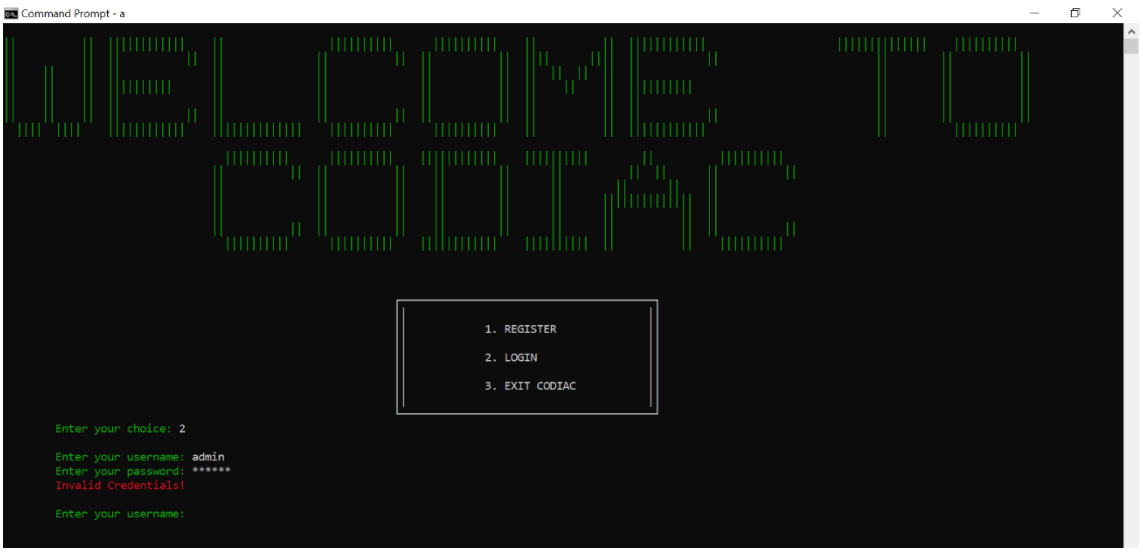


4.2. ADMIN TEST CASE RESULTS

Test Case 16:

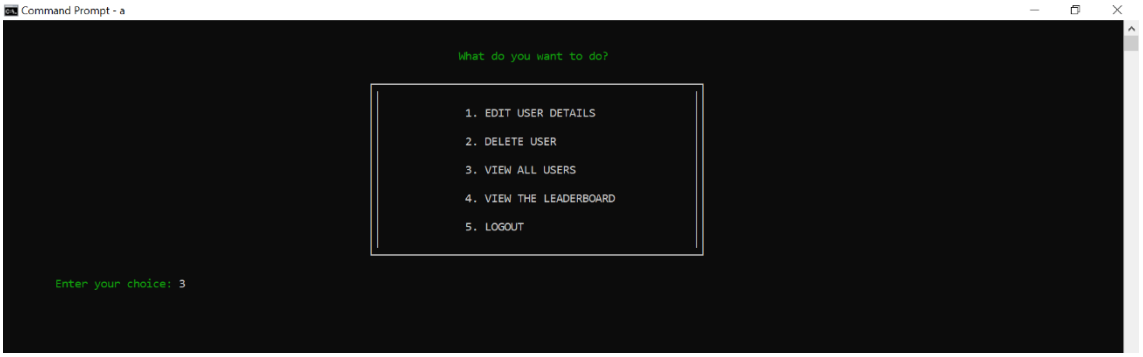


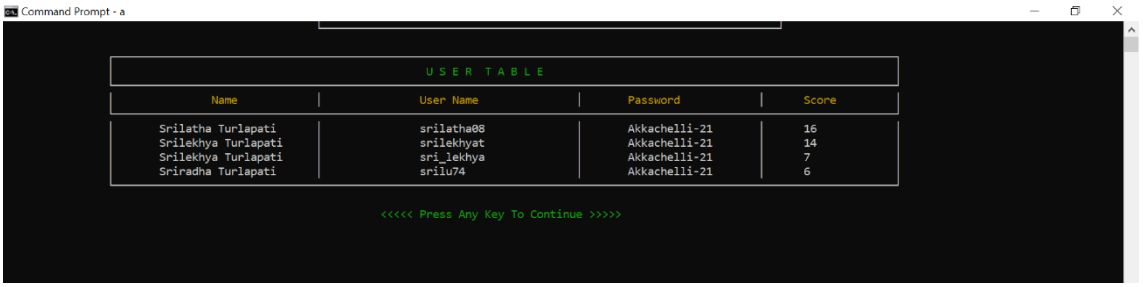
Test Case 17:



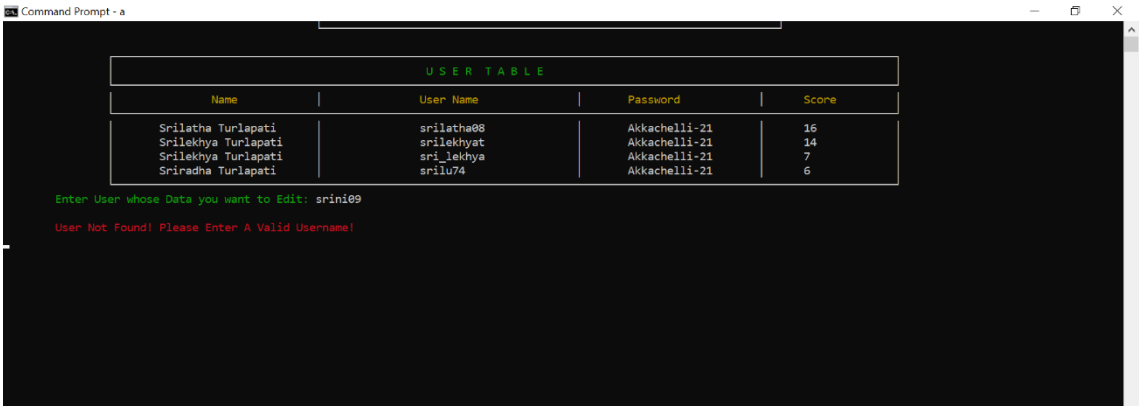
Test Case 18:

Test Case 19:

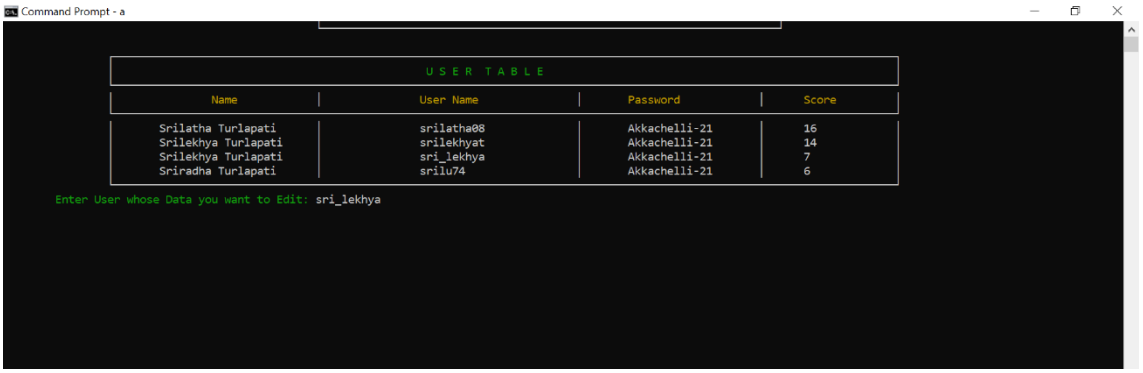


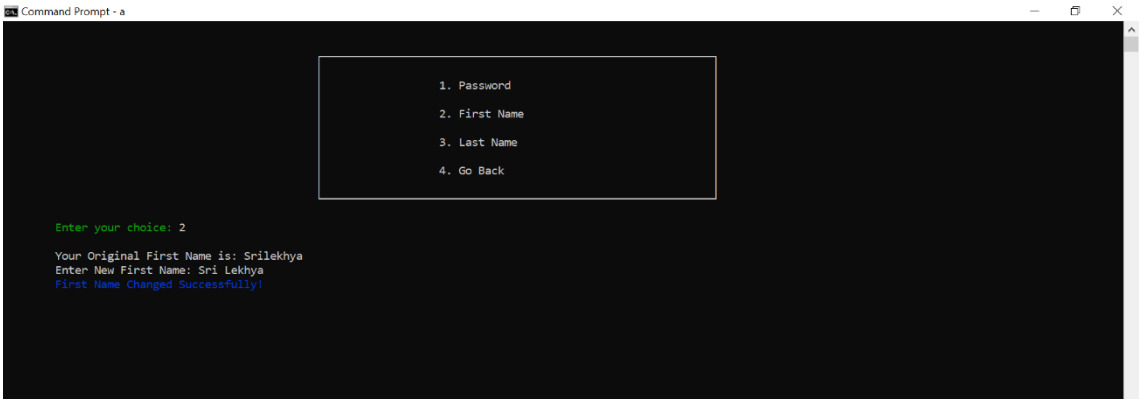


Test Case 20:

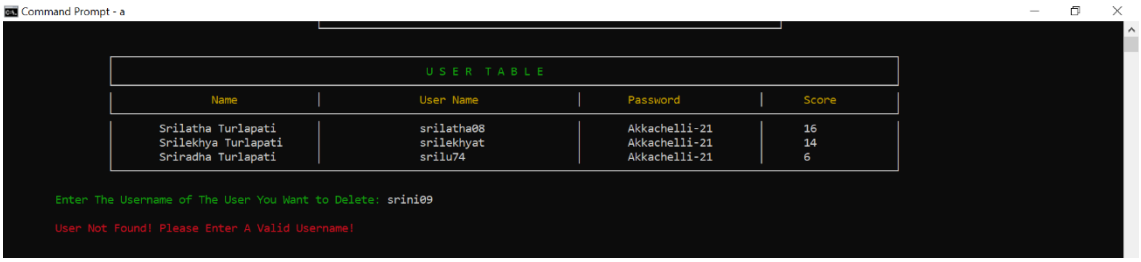


Test Case 21:

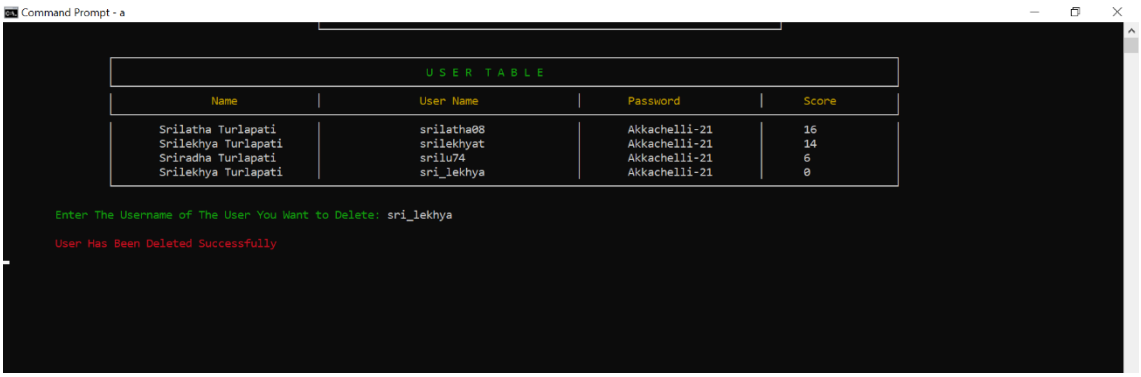




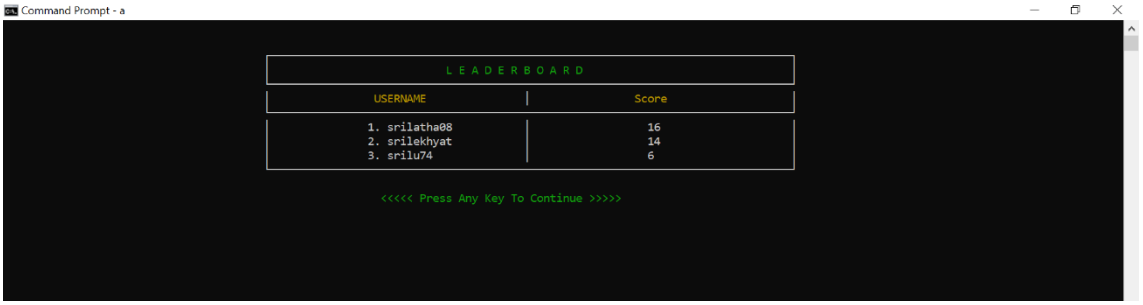
Test Case 22:



Test Case 23:



Test Case 24:



Test Case 25:

5. ADDITIONAL KNOWLEDGE ACQUIRED

Implementing this project in C Language has introduced us to different libraries such as: ‘conio.h’, ‘time.h’ and ‘windows.h’. We were able to use the knowledge we have on the Linked List Data Structure and execute it as a real-time application. We used the ‘windows.h’ library for controlling the display colours in a controlled manner. We explored the ‘time.h’ and ‘conio.h’ libraries for achieving a look-and-feel of an actual window application by constructing our own time delay function.

Also, we have further improved our knowledge in file-handling because of the vast amount of data manipulation we have done using text files.

Other than this, we have learnt the value of team spirit and have understood the intention behind working in teams. We have learnt to be team players.

6. CONCLUSION AND FUTURE WORK

To conclude, we have built a platform in which users can improve their skill-set in important Computer Science-based concepts such as C, Python and OOPs. The intention behind this project was to enhance our knowledge in these crucial subjects and further provide the same to all users of our platform. We also wanted to inculcate the practise of Self-Learning, that is, without guidance of professors or institutions. Our motive is for students to take ownership of their learning and additionally build independence.

Our future work includes incorporating more concepts such as Data Structures in our Learning and Quiz modules. Also, we would like to include a feature which allows the User to send requests to the Admin for any changes they require. Other than this, we are deliberating on increasing the authority of the Admin by shifting some features from read-only to editable access as well.

This project can be further improved by converting it into a Web Application using Python and the Django Framework or a Mobile Application using Flutter or React Native.

7. REFERENCES

C Language Documentation (For referring C libraries): https://devdocs.io/c/

Stack Overflow (For Debugging Errors): https://stackoverflow.com/

Reference for Setting Colours in Console: https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/color