

**QUIZ GAME PROJECT**

**Project Team Members :**

1. **Aadil**
2. **Anirudh P K**
3. **Albin Reji**
4. **Gautham Varma**

**CODE SUMMARY**

**Quiz Game**

The quiz game has several features, including user registration, login, different quiz categories (e.g., football, general knowledge, sports, TV shows and movies, music, geography, and cricket), two rounds (warmup and challenge), and a scoring system. Here's a brief description of the game:

This C program implements a quiz game with different categories of questions. It begins by presenting a main menu with options to start the game, view the highest score, reset the score, get help, register as a new user, or quit the game. Users can register with a username and password and log in to play the game.

The game consists of two rounds: a warm-up round and a challenge round. In the warm-up round, the player needs to answer at least 2 out of 3 questions correctly to qualify for the challenge round. In the challenge round, the player can choose from various categories (e.g., football, general knowledge, etc.) and answer 10 questions. Each correct answer is awarded $100,000, and the player's score is accumulated.

After completing the challenge round, the player's total score is displayed, and they have the option to play another game or return to the main menu. If the player scores $1,000,000, they become a millionaire.

The game also keeps track of the highest score and allows users to reset their score. User data is stored in a file ("users.txt"), and the highest score is stored in another file ("score.txt").

Overall, this C program provides an interactive quiz game with multiple features and categories to test the player's knowledge and win virtual money.

This code is a C program for a text-based quiz game. Here's a summary of its main features:

1. **Header Includes**:

- The code includes standard C libraries such as `stdio.h`, `conio.h`, `ctype.h`, `stdlib.h`, and `string.h` for various functionalities.

2. **Constants:**

- Several constants are defined using `#define` directives to specify the maximum number of questions, users, and the maximum lengths for usernames and passwords.

3**. Structs:**

- Two structs are defined: `User` and `Question`. `User` stores user information with a username and password, while `Question` represents a quiz question with question text, answer options, and the correct option.

4. **Functions:**

- The code defines various functions, including:

- `read\_questions`: Reads questions from a file into an array of `Question` structs.

- `show\_record`: Displays the highest score recorded in a file named "score.txt."

- `reset\_score`: Resets the highest score to zero in the "score.txt" file.

- `help`: Provides instructions and help for the game.

- `edit\_score`: Updates the highest score if the player's score is higher.

- `shuffle\_questions`: Shuffles the order of questions in an array.

- `register\_user`: Allows users to register by providing a username and password.

- `login`: Allows registered users to log in by verifying their credentials.

- `main`: The main function serves as the game controller, presenting a menu and handling user choices to navigate through the game.

5. **Main Function:**

- The `main` function is the entry point and controls the game flow.

- It presents a menu with options like starting the game, viewing the highest score, resetting the score, getting help, registering a new user, and quitting.

- Depending on the user's choice, it directs the program to various sections of the game, including warm-up and challenge rounds for different quiz categories.

- It keeps track of the player's score and displays the prize amount if the player wins.

- The program also offers the option to play another round or return to the main menu.

6. **File Handling:**

- The code reads and writes user information and questions from/to text files (e.g., "users.txt," "questions.txt") to maintain data persistence.

7. **Flow Control:**

- The code uses control structures like loops (e.g., `for`, `while`) and conditionals (e.g., `if`, `else`) to manage the flow of the game, validate user input, and display appropriate information.

8. User Interaction:

- The game interacts with the user through the console, displaying questions, options, and instructions.

9**. Error Handling:**

- The code includes basic error handling, such as checking if files can be opened successfully and handling invalid user input.

Overall, this code implements a quiz game that allows users to register, log in, answer quiz questions, and compete for high scores. It demonstrates file handling, user input validation, and game control flow within a text-based interface.

The working of this code can be summarized as follows:

1. **Initialization and Menu Display:**

- The program starts by displaying a main menu that offers several options: starting the game, viewing the highest score, resetting the score, getting help, registering a new user, and quitting.

- The user selects an option by entering a corresponding key.

2. **User Registration and Login**:

- If the user chooses to register, they can provide a username and password. The program checks if the username already exists and registers the user if it's unique.

- If the user selects login, they enter their username and password, and the program checks the credentials against existing user data to allow access.

3. **Warm-up Round:**

- If the user successfully logs in or registers and qualifies (by answering at least 2 out of 3 warm-up questions correctly), they proceed to the challenge round.

- The warm-up round consists of 3 general knowledge questions, and the user's answers are evaluated.

4. **Challenge Round Selection:**

- In the challenge round, the user can select a category of questions (e.g., football, general knowledge, sports) by choosing a corresponding number.

- Depending on the category selected, the program loads questions from a specific file and presents a series of 10 questions to the user.

5. **Question Presentation and Answer Evaluation**:

- The program displays each question along with four answer options (A, B, C, D).

- The user selects an option, and the program checks if it matches the correct answer for that question.

- Correct answers earn the user $100,000 for each question, and the score is updated accordingly.

6**. Score Calculation and Display:**

- The program calculates the user's total score and displays it at the end of the challenge round.

- If the user's score is between $10 and $1,000,000, they receive a congratulatory message and the prize amount.

- If the user's score is $1,000,000, they are declared a millionaire.

7**. Options to Continue or Quit:**

- After the challenge round, the user has the option to play another round by selecting "Y" or return to the main menu by selecting any other key.

- If the user chooses to play another round, they can select a different category.

- If the user quits, the program allows them to edit their highest score if they achieved a new high score during the game.

8**. File Handling:**

- The program uses text files ("users.txt" and "score.txt") to store user data and the highest score, allowing data persistence across different runs of the program.

9. **Error Handling and Help:**

- The code includes basic error handling to check if files can be opened and handles invalid user input.

- The "help" option provides instructions and tips for playing the game.

Overall, this code creates an interactive quiz game with options for user registration, login, quiz rounds, scoring, and data persistence. It offers a text-based interface for users to enjoy the game and compete for high scores.

**Code :**

#include<stdio.h>

#include<conio.h>

#include<ctype.h>

#include<stdlib.h>

#include<string.h>

#define MAX\_QUESTIONS 20

#define MAX\_USERS 100

#define MAX\_NAME\_LENGTH 20

#define MAX\_PASSWORD\_LENGTH 30

struct User {

char username[MAX\_NAME\_LENGTH];

char password[MAX\_PASSWORD\_LENGTH];

};

struct Question {

char question\_text[500];

char options[4][100];

char correct\_option;

};

void read\_questions(FILE \*file, struct Question \*questions, int num\_questions) {

int i,j;

for ( i = 0; i < num\_questions; i++) {

fscanf(file, "%499[^\n]\n", questions[i].question\_text);

for ( j = 0; j < 4; j++) {

fscanf(file, "%99[^\n]\n", questions[i].options[j]);

}

fscanf(file, "%c\n", &questions[i].correct\_option);

}

}

void show\_record() {

FILE \*f = fopen("score.txt", "r");

if (f == NULL) {

printf("No high score available.\n");

return;

}

char name[MAX\_NAME\_LENGTH];

float score;

fscanf(f, "%s%f", name, &score);

printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\t\t %s has secured the Highest Score %.2f", name, score);

printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

fclose(f);

getchar();

}

void reset\_score() {

FILE \*f = fopen("score.txt", "w");

if (f == NULL) {

printf("Error opening score file.\n");

return;

}

fprintf(f, "%s %.2f", "NoName", 0.0);

fclose(f);

printf("\n\n\t\tScore is reset to zero.\n");

}

void help()

{system("cls");

printf("\n\n HELP");

printf("\n -------------------------------------------------------------------------");

printf("\n ......................... C program Quiz Game...........");

printf("\n >> There are two rounds in the game, WARMUP ROUND & CHALLANGE ROUND");

printf("\n >> In warmup round you will be asked a total of 3 questions to test your general");

printf("\n knowledge. You will be eligible to play the game if you can give atleast 2");

printf("\n right answers otherwise you can't play the Game...........");

printf("\n >> Your game starts with the CHALLANGE ROUND. In this round you will be asked");

printf("\n total 10 questions each right answer will be awarded $100,000.");

printf("\n By this way you can win upto ONE MILLION cash prize in USD...............");

printf("\n >> You will be given 4 options and you have to press A, B ,C or D for the");

printf("\n right option");

printf("\n >> You will be asked questions continuously if you keep giving the right answers.");

printf("\n >> No negative marking for wrong answers");

printf("\n\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BEST OF LUCK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\t\*\*\*\*\*C PROGRAM QUIZ GAME is developed by Technical Team 5\*\*\*\*\*\*\*\*");}

void edit\_score(float score, char plnm[20]) {

system("cls");

float sc;

char nm[20];

FILE \*f;

f = fopen("score.txt", "r");

fscanf(f, "%s%f", nm, &sc); // Corrected line

if (score >= sc) {

sc = score;

fclose(f);

f = fopen("score.txt", "w");

fprintf(f, "%s\n%.2f", plnm, sc);

fclose(f);

}

}

void shuffle\_questions(struct Question arr[], int n) {

int i,j;

srand(time(NULL));

for (i = n - 1; i > 0; i--) {

j = rand() % (i + 1);

struct Question temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

void register\_user() {

struct User users[MAX\_USERS];

int num\_users = 0;

int i;

// Read existing user data from file

FILE \*users\_file = fopen("users.txt", "r");

if (users\_file) {

while (fscanf(users\_file, "%s %s", users[num\_users].username, users[num\_users].password) != EOF) {

num\_users++;

}

fclose(users\_file);

}

// Register a new user

char new\_username[MAX\_NAME\_LENGTH];

char new\_password[MAX\_PASSWORD\_LENGTH];

printf("\nEnter a new username: ");

scanf("%s", new\_username);

// Check if the username already exists

int username\_exists = 0;

for (i = 0; i < num\_users; i++) {

if (strcmp(new\_username, users[i].username) == 0) {

username\_exists = 1;

break;

}

}

if (username\_exists) {

printf("Username already exists. Registration failed.\n");

}

else {

printf("Enter a password: ");

scanf("%s", new\_password);

// Add the new user to the array

strcpy(users[num\_users].username, new\_username);

strcpy(users[num\_users].password, new\_password);

num\_users++;

// Write the updated user data back to the file

users\_file = fopen("users.txt", "w");

for (i = 0; i < num\_users; i++) {

fprintf(users\_file, "%s %s\n", users[i].username, users[i].password);

}

fclose(users\_file);

printf("User registered successfully!\n");

getch();

}

}

char player\_name[MAX\_NAME\_LENGTH];

int login() {

struct User users[MAX\_USERS];

int num\_users = 0;

int i;

// Read user data from file

FILE \*users\_file = fopen("users.txt", "r");

if (users\_file) {

while (fscanf(users\_file, "%s %s", users[num\_users].username, users[num\_users].password) != EOF) {

num\_users++;

}

fclose(users\_file);

}

// Prompt for login credentials

char entered\_username[MAX\_NAME\_LENGTH];

char entered\_password[MAX\_PASSWORD\_LENGTH];

printf("LOGIN");

printf("\nEnter your username: ");

scanf("%s", entered\_username);

printf("Enter your password: ");

scanf("%s", entered\_password);

// Check if the entered credentials match any user

for (i = 0; i < num\_users; i++) {

if (strcmp(entered\_username, users[i].username) == 0 && strcmp(entered\_password, users[i].password) == 0) {

printf("\n\n LOGIN SUCCESSFUL");

strcpy(player\_name, entered\_username);

return 1; // Login successful

}

}

return 0; // Login failed

}

int main()

{

int count = 0;

int i, n,ch;

float score;

char choice;

struct Question questions[15];

struct Question football[MAX\_QUESTIONS];

struct Question gk[MAX\_QUESTIONS];

struct Question sports[MAX\_QUESTIONS];

struct Question TV[MAX\_QUESTIONS];

struct Question music[MAX\_QUESTIONS];

struct Question geography[MAX\_QUESTIONS];

struct Question cricket[MAX\_QUESTIONS];

FILE \*questions\_file= fopen("questions.txt","r");

FILE \*football\_file = fopen("football.txt", "r");

FILE \*gk\_file = fopen("gk.txt", "r");

FILE \*sports\_file = fopen("sports.txt", "r");

FILE \*TV\_file=fopen("TV.txt","r");

FILE \*music\_file=fopen("music.txt","r");

FILE \*geography\_file=fopen("geography.txt","r");

FILE \*cricket\_file=fopen("cricket.txt","r");

if (football\_file && gk\_file && sports\_file && TV\_file && music\_file && geography\_file && cricket\_file) {

// Read questions from files into arrays

read\_questions(questions\_file, questions, 15);

read\_questions(football\_file, football, MAX\_QUESTIONS);

read\_questions(gk\_file, gk, MAX\_QUESTIONS);

read\_questions(sports\_file, sports, MAX\_QUESTIONS);

read\_questions(TV\_file, TV, MAX\_QUESTIONS);

read\_questions(music\_file, music, MAX\_QUESTIONS);

read\_questions(geography\_file, geography, MAX\_QUESTIONS);

read\_questions(cricket\_file, cricket, MAX\_QUESTIONS);

// Close the question files

fclose(questions\_file);

fclose(football\_file);

fclose(gk\_file);

fclose(sports\_file);

fclose(TV\_file);

fclose(music\_file);

fclose(geography\_file);

fclose(cricket\_file);

} else {

printf("Error opening question files.\n");

return 1;

}

mainhome:

printf("\n\n\t\t\tC PROGRAM QUIZ GAME");

printf("\n\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n\t\t\t WELCOME ");

printf("\n\t\t\t to ");

printf("\n\t\t\t THE GAME ");

printf("\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n\n\t\t > Press S to start the game");

printf("\n\t\t > Press V to view the highest score");

printf("\n\t\t > Press R to reset score");

printf("\n\t\t > Press H for help");

printf("\n\t\t > Press N to register");

printf("\n\t\t > Press Q to quit");

printf("\n\t\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\t\t ");

choice = toupper(getch());

if (choice == 'V') {

show\_record();

goto mainhome;

}

else if (choice == 'H') {

help();

getch();

goto mainhome;

}

else if (choice == 'R') {

reset\_score();

printf("\n\n\t\tScore is reset to zero.");

getch();

goto mainhome;

}

else if (choice == 'Q')

return 0;

else if (choice == 'N') {

printf("REGISTRATION");

register\_user();

goto mainhome;

}

else if (choice == 'S') {

if (login()) {

goto start;

} else {

printf("\nLogin failed. Press any key to return to the main menu.");

getch();

goto mainhome;

}

start:

system("cls");

printf("\n ------------------ Welcome %s to C Program Quiz Game --------------------------", player\_name);

printf("\n\n Here are some tips you might wanna know before playing:");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n >> There are two rounds in this Quiz Game, WARMUP ROUND & CHALLANGE ROUND");

printf("\n >> In warmup round, you will be asked a total of 3 questions to test your general knowledge.");

printf("\n You need to answer at least 2 questions correctly to qualify for the next round.");

printf("\n >> Your game starts with the CHALLANGE ROUND. In this round, you will be asked a total of 10 questions.");

printf("\n Each correct answer will be awarded $100,000.");

printf("\n By this way you can win up to ONE MILLION cash prize in USD.");

printf("\n >> You will be given 4 options and you have to press A, B, C or D for the right option.");

printf("\n >> You will be asked questions continuously as long as you keep giving the right answers.");

printf("\n >> There is no negative marking for wrong answers.");

printf("\n\n\t\t!!!!!!!!!!!!! BEST OF LUCK !!!!!!!!!!!!!");

printf("\n\n\n Press Y if you are ready to start the game!");

printf("\n Press any other key to return to the main menu!");

if (toupper(getch()) == 'Y') {

goto home;

} else {

goto mainhome;

}

home:

system("cls");

score = 0;

printf("\n ------------------ WARMUP ROUND --------------------------");

shuffle\_questions(questions, sizeof(questions) / sizeof(questions[0]));

for (i = 0; i < 3; i++) {

printf("\n\n%s\n", questions[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", questions[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == questions[i].correct\_option) {

printf("\n\nCorrect!");

count += 1;

getch();

} else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", questions[i].correct\_option);

getch();

}

}

if(count>=2)

{

goto test;}

else

{

system("cls");

printf("\n\nSORRY YOU ARE NOT ELIGIBLE TO PLAY THIS GAME, BETTER LUCK NEXT TIME");

getch();

goto mainhome;

}

test:

system("cls");

printf("\n\n\t\*\*\* CONGRATULATION %s you are eligible to play the Game \*\*\*",player\_name);

printf("\n\n\n\n\t!Press any key to Start the Game!");

if(toupper(getch())=='p')

{goto game;}

game:

system("cls");

printf("\n ------------------ CHALLENGE ROUND --------------------------");

printf("\n\t\tEnter your choice of Interest\n\n");

printf("\n\t\t1. Football\n\n");

printf("\n\t\t2. General Knowledge\n\n");

printf("\n\t\t3. Sports\n\n");

printf("\n\t\t4. TV Shows and Movies\n\n");

printf("\n\t\t5. Music or Song\n\n");

printf("\n\t\t6. Geography\n\n");

printf("\n\t\t7. Cricket\n\n");

scanf("%d",&ch);

if(ch==1){

system("cls");

score = 0;

shuffle\_questions(football, sizeof(football) / sizeof(football[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", football[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", football[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == football[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", football[i].correct\_option);

getch();

}

}

}

else if(ch==2){

system("cls");

score = 0;

shuffle\_questions(gk, sizeof(gk) / sizeof(gk[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", gk[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", gk[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == gk[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", gk[i].correct\_option);

getch();

}

}

}

else if(ch==3){

system("cls");

score = 0;

shuffle\_questions(sports, sizeof(sports) / sizeof(sports[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", sports[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", sports[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == sports[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", sports[i].correct\_option);

getch();

}

}

}

else if(ch==4){

system("cls");

score = 0;

shuffle\_questions(TV, sizeof(TV) / sizeof(TV[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", TV[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", TV[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == TV[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", TV[i].correct\_option);

getch();

}

}

}

else if(ch==5){

system("cls");

score = 0;

shuffle\_questions(music, sizeof(music) / sizeof(music[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", music[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", music[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == music[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", music[i].correct\_option);

getch();

}

}

}

else if(ch==6){

system("cls");

score = 0;

shuffle\_questions(geography, sizeof(geography) / sizeof(geography[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", geography[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", geography[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == geography[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", geography[i].correct\_option);

getch();

}

}

}

else if(ch==7){

system("cls");

score = 0;

shuffle\_questions(cricket, sizeof(cricket) / sizeof(cricket[0]));

for (i = 0; i < 10; i++) {

printf("\n\n%s\n", cricket[i].question\_text);

for (n = 0; n < 4; n++) {

printf("\n\t%s", cricket[i].options[n]);

}

printf("\n\n\n");

choice = toupper(getch());

if (choice == cricket[i].correct\_option) {

printf("\n\nCorrect!");

score += 100000;

getch();

}

else {

printf("\n\nWrong!");

printf("\n\nThe correct option is %c", cricket[i].correct\_option);

getch();

}

}

}

else{

printf("\n\t\tINVALID OPTION\n\t\tEXITING!!!");

goto mainhome;

}

goto score;

score:

system("cls");

if(score>10.00 && score<1000000.00)

{

printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONGRATULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t You won $%.2f",score);goto go;}

else if(score==1000000.00)

{

printf("\n\n\n \t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONGRATULATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\t\t YOU ARE A MILLIONAIRE!!!!!!!!!");

printf("\n\t\t You won $%.2f",score);

printf("\t\t Thank You!!");

}

else

{

printf("\n\n\t\*\*\*\*\*\*\*\* SORRY YOU DIDN'T WIN ANY CASH \*\*\*\*\*\*\*\*");

printf("\n\t\t Thanks for your participation");

printf("\n\t\t TRY AGAIN");goto go;}

go:

puts("\n\n Press Y if you want to play next game");

puts(" Press any key if you want to go main menu");

if (toupper(getch())=='Y')

goto home;

else

{

edit\_score(score,player\_name);

goto mainhome;

}

}

else{

printf("Invalid choice");

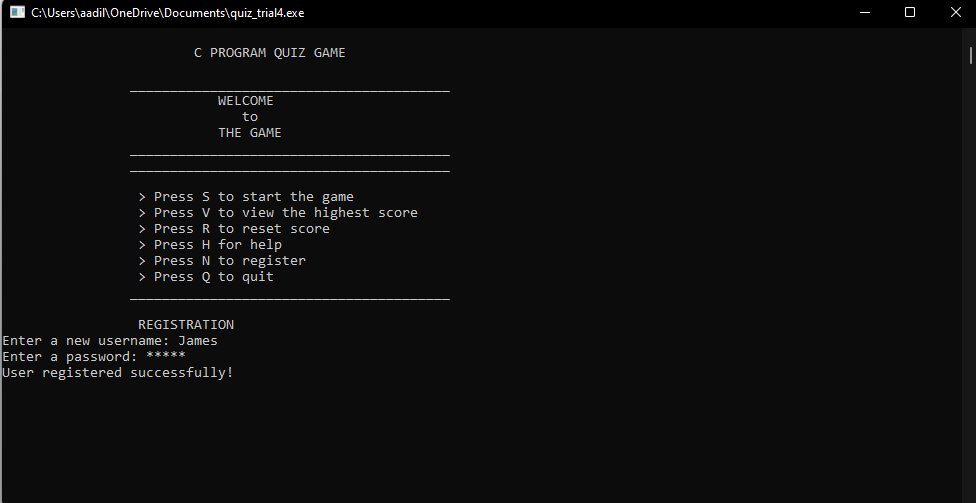
goto mainhome;

}

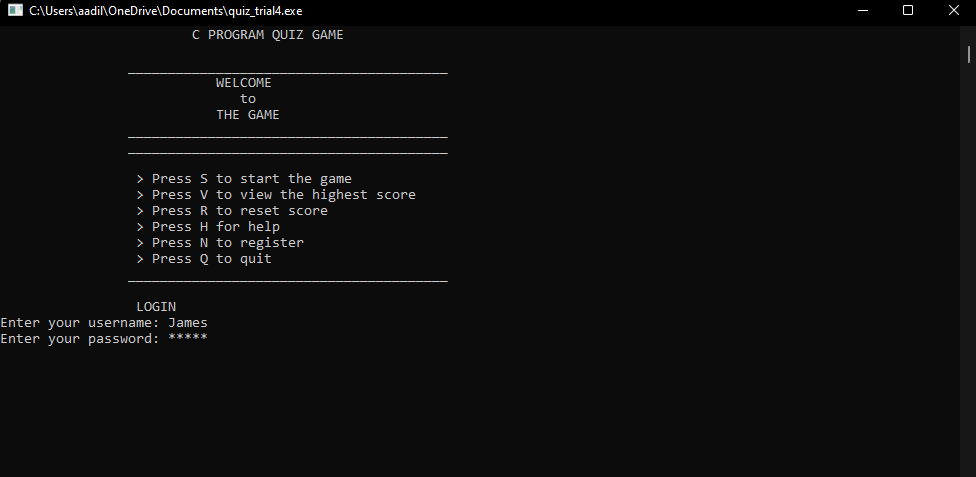
}

**Output :**

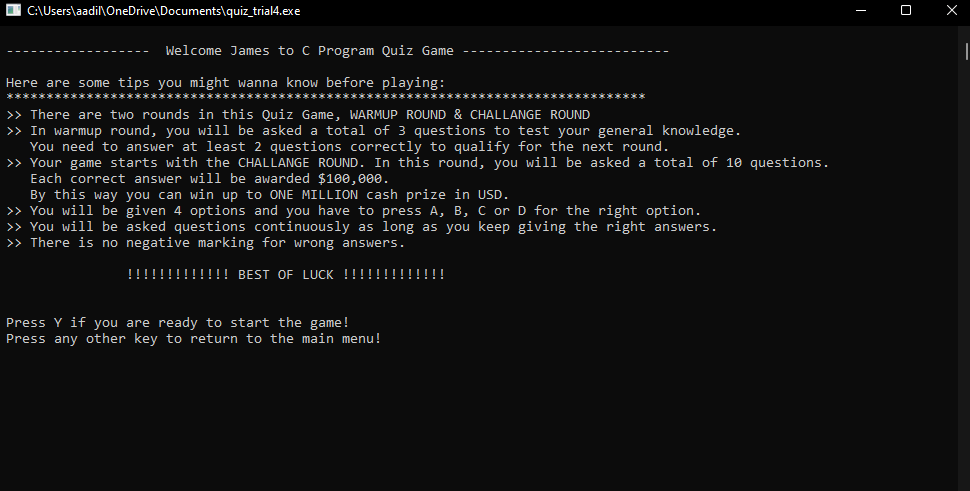
**1 ) registration page**

****

**2) Login page :**

****

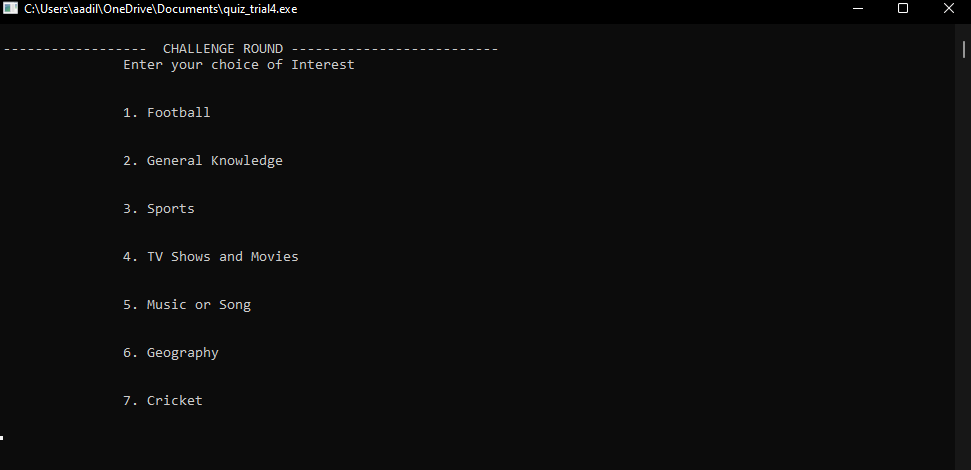
**3) Tips and instruction :**

****

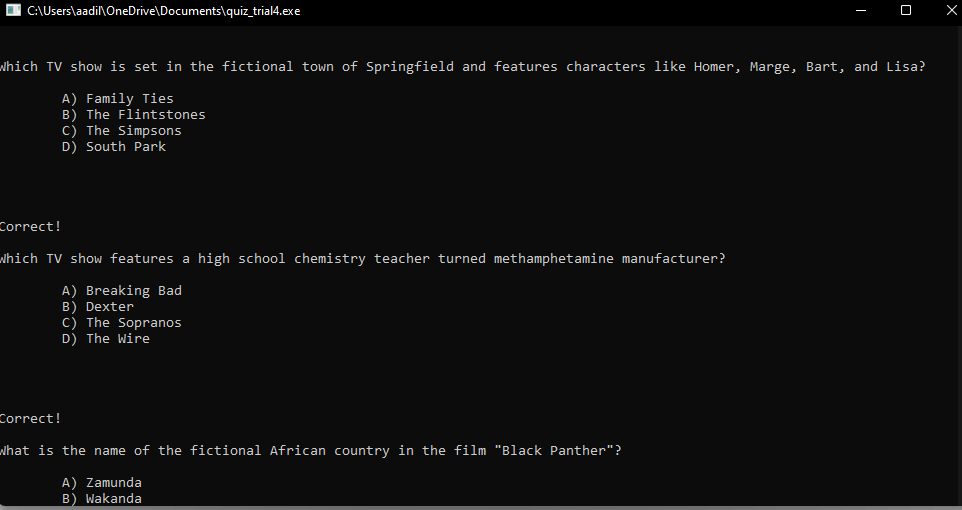
**4) Warmup round:**

****

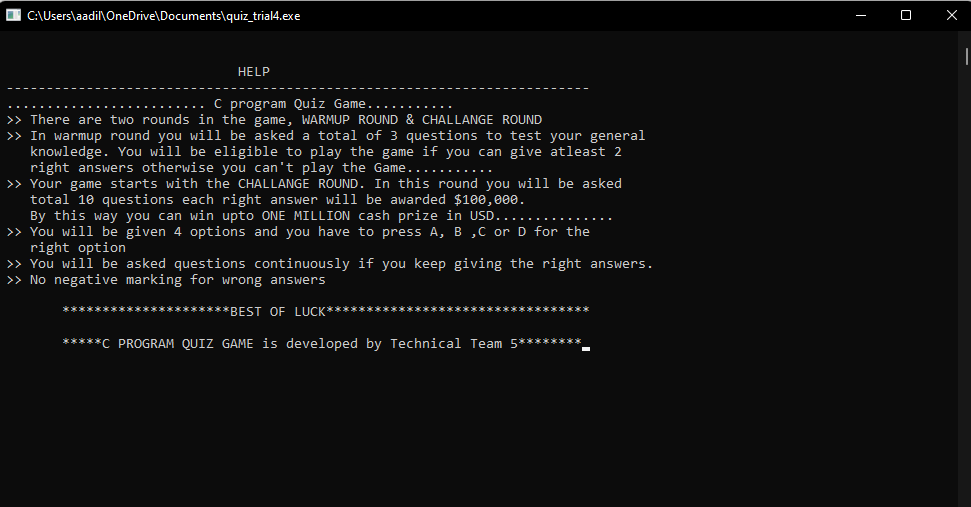
**5) Choose the intrested topic:**

****

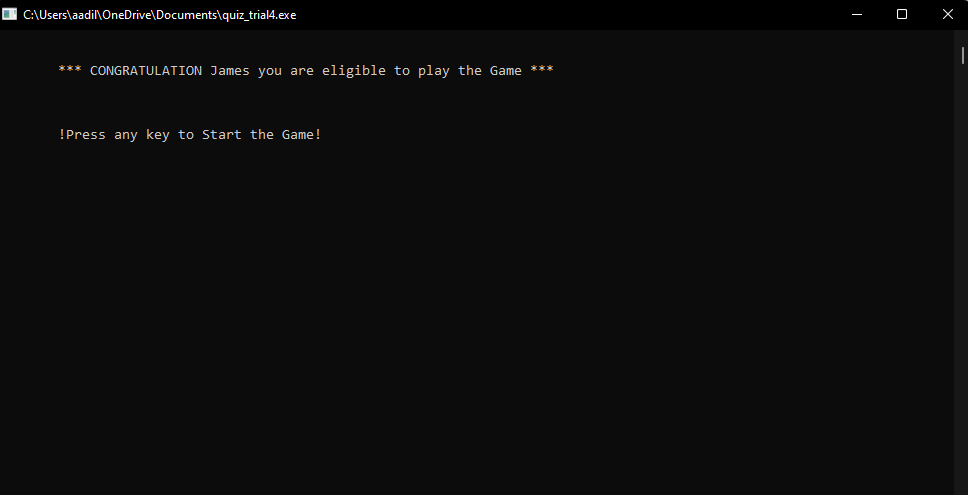
**6) Questions**

****

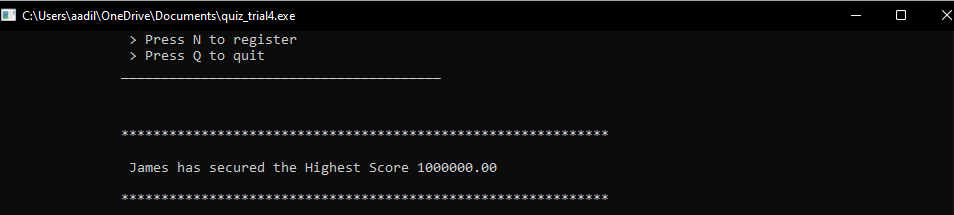
**7) help**

****

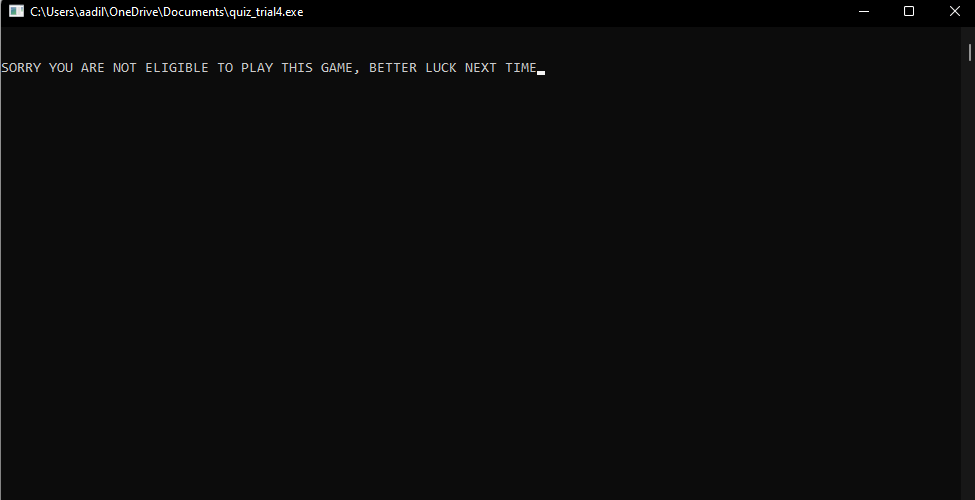
**8) if the player is eligible**

****

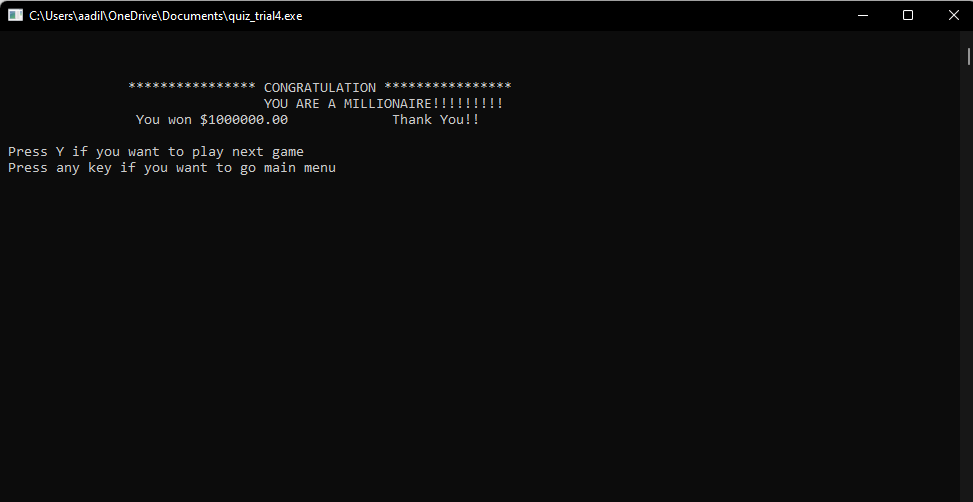
**9) Display of high score:**

****

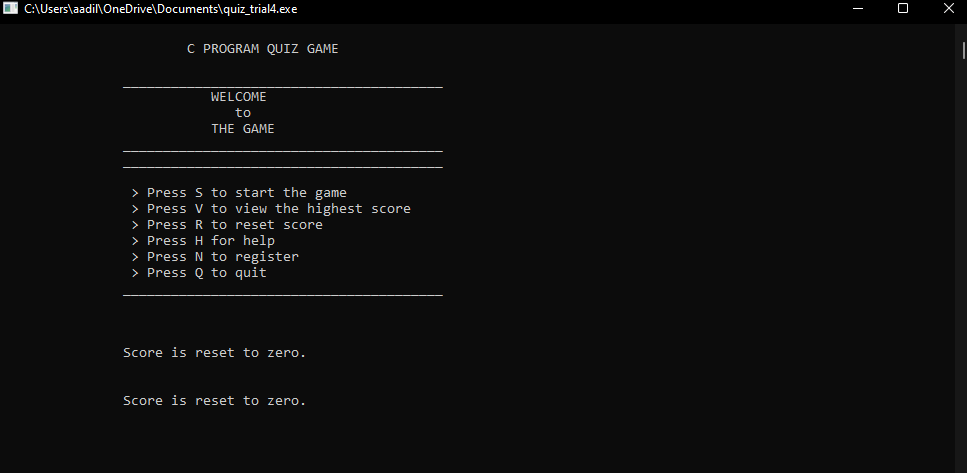
**10) If the player is not eligible:**

****

**11) if player won the game:**

****

**12) Start a new game and reset the score:**

****