Iyad Ahmad Farooq

231006436

The Scholar & The Obsidian Raven

The Knowledge Comeback

Brief Introduction: -

The Idea:

This is a Quiz Game where you answer many questions including various topics such as Politics, Sports, Videogames, Movies, Geography, etc. . . .

The Story:

You were a knowledgeable scholar, you had answers for questions that no one even thought of.

But as your knowledge circle widened, so did your enemies' numbers.

One of them being

**The Obsidian Raven!**

He hunts in the shadows, so when you were expanding your knowledge late at night in the library while everyone was sleeping.

He took the chance and struck you.

With his powerful curse, he was able to steal all your knowledge.

The knowledge he could not steal was who you were.

Now, the only way to get your knowledge back is to earn it from him again by answering all of his questions, which he will ask using all of the knowledge he stole.

Good luck, traveler :)

Basic Functionality:

The user puts his answer and the program compares it to the correct answers, there are some counter measures that we are going to discuss that eliminate the margin of error of the user inputting a wrong input

The Code: -

Part 1:

This is just your basic libraries; it has functions that we are going to need later on

Part 5:

Just some declarations for variables we are going to use later on

Part 2:

This a struct that decided the question-and-answer format of our database later, basically it’s like having a paper that has 2 spaces for you to write your question and answer, we are not going to input the questions manually however we are going to use a file.

Part 6:

Here we have a part that reads the questions from a txt file we called questions, first we created a pointer of FILE type since we are dealing with files and named it Database, in case that this file does not exist it’s going to warn you about it, assuming the file exists, it is going to read 99 characters maximum until it sees a (,), that would be our questions, after it is going to do the same until it sees (\n), that would be our correct answer, it is going to save all of that information in an array of structs we created previously called Questions, after we are done we close the file and use a function we created called shuffle\_questions, from the name it shuffles the questions to add a bit more of a challenge

Part 4:

Here is our shuffle function, we are going to need our <stdlib.h> library for this one in order to use srand() function and rand() function, in order to start the randomization itself we are going to srand(time(NULL)) after that we will create a loop that goes through all the questions and shuffle by creating and integer (in that case int j) which will be assigned a random number from 0 to the maximum number of questions, then it is just a case of a regular swap

Part 7:  
The game starts by giving you the option to either read the story or not, if choose to go on and read the story you will get a paragraph that explains the story very well, and 30 seconds to read the it before starting.

In both cases you will move on to the main game, you will get a timer of 1 minutes to answer each question, a \r is added to ensure that it won’t print out each line in a new line but rather replace the old line, if the user decided to answer a function called (\_kbhit()) will be activated which will take the user’s answer and activate answered to true so it doesn’t activate the timer message, there is also a Sleep() function in order not to put load on the CPU

Part 8:

In this part after we have taken the user’s answer we will check if there are any errors by printing “Unknown Error” if its value is NULL, this is more for the developers to check what is the problem.

Incase everything goes smoothly, no we are not going to check the answer yet, first we need to take in mind that the user might enter the answer in caps, so we are going to use a function called tolower() which is under the <ctype.h> library, we are going to put in a loop and it will go through each character in the user answer and make it lowercase so when we compare the answers it matches the format we are using;

(The Question, Answer)

After that, we need to solve a different problem, when the user puts in an answer it takes the enter as a character in the answer as well, so when we are comparing answer, we are essentially doing   
  
strcmp(Answer\n, correct answer)  
  
it is always going to be false

In order to solve that we are using strcspn(), which in short searches for a character that you choose, finds it, and then replace it with whatever you want, in our case, we are going to use \0 which is a null terminator so we can compare the answers correctly.

Ok there is another problem, there are about 265 Questions, We need a way to exit this for some random reason, so if at any points the user inputs “end game” it will ask him if he is sure and then if he presses Y or y for yes basically it will give the user his score and close the program, the reason for the sleep is because when we use the .exe version the program closes immediately and we need to give the user a second to read his score, in case he chooses n or N for No it will just exit the round and ask him if he wants to play another round

Part 9:

This part just tells you if your answer is correct or not and adds to the user score to print for later

Part 10:

This part is the end of the program . . . if the player chooses to, incase if the player ends the round or finishes the questions he will be asked if he wants to try again or not, same thing Y or y for Yes and n or N for No, the idea here is the whole code is wrapped in a do while loop where the code executes one time and then the condition that keeps the loop going can be either broken by the player closing down the game, or keeping it the same and going on again, if the player choose N or n for No he will get message and a few seconds to read it, this part does not serve and important functionality to the code, it is more of just something to keep the player engaged about the game, wanting to come back again, something to remotivate him for coming back again and retrying

Part 3:

This a function that decided the end message of the user based on his score, as you can see different gradings for different answered question correctly, I decided that it is better to put it in a function since as you can see the code for it is long since it is mostly reported plus we are going to use a lot in our main function

The Code

///////////////////////////////////////Part 1/////////////////////////////////////////////////////

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <ctype.h>

#include <unistd.h>

#include <conio.h>

#include <windows.h>

#include <stdbool.h>

///////////////////////////////////////Part 2/////////////////////////////////////////////////////

struct Question\_Format {

char Questions[100];

char Answers[100];

};

////////////////////////////////////////////Part 3////////////////////////////////////////////////

void Destiny(int user\_score, int Question\_Number)

{

if (user\_score == 265)

{

printf("%d/%d Perfect! You've fully reclaimed your knowledge. You are victorious :D\n", user\_score, Question\_Number);

}

else if (user\_score < 261 && user\_score > 236)

{

printf("%d/%d Wow! You're almost there! You're on the brink of victory! :D\n", user\_score, Question\_Number);

}

else if (user\_score < 235 && user\_score > 210)

{

printf("%d/%d Good, but not great! You can do better! :)\n", user\_score, Question\_Number);

}

else if (user\_score < 209 && user\_score > 158)

{

printf("%d/%d Good performance, but not enough still. Keep pushing and you will get there :)\n", user\_score, Question\_Number);

}

else if (user\_score < 157 && user\_score > 132)

{

printf("%d/%d Close one, you need to focus more >:)\n", user\_score, Question\_Number);

}

else if (user\_score < 131 && user\_score > 106)

{

printf("%d/%d You are wounded. Stand back and heal yourself - O -\n", user\_score, Question\_Number);

}

else if (user\_score < 105 && user\_score > 81)

{

printf("%d/%d You barely got out of there alive :O\n", user\_score, Question\_Number);

}

else if (user\_score < 81 && user\_score > 56)

{

printf("%d/%d You are not ready. You need more training -\_- \n", user\_score, Question\_Number);

}

else if (user\_score < 56 && user\_score >= 0)

{

printf("%d/%d You fell under the Raven's mind-control curse :X\n", user\_score, Question\_Number);

}

}

////////////////////////////////////////////Part 4//////////////////////////////////////////////////

void shuffle\_questions(struct Question\_Format Questions[], int Question\_Number)

{

srand(time(NULL));

for(int i = Question\_Number; i >= 0; i--)

{

int j = rand() % Question\_Number;

struct Question\_Format temp = Questions[i];

Questions[i] = Questions[j];

Questions[j] = temp;

}

}

////////////////////////////////////////////Part 5//////////////////////////////////////////////////

int main()

{

char Try\_Again = 'Y';

do

{

int user\_score = 0, Question\_Number = 0, number\_of\_questions = 0;

struct Question\_Format Questions[300];

char user\_answer[100], program\_ender[] = "end game", Try\_Again\_2 = 'L', read\_or\_not = 'S';

bool answered = true;

//////////////////////////////////////////////Part 6///////////////////////////////////////////////

FILE \*Database = fopen("Question.txt", "r");

if(Database == NULL)

{

printf("This File Does not exist");

return 1;

}

while(fscanf(Database, "%99[^,],%99[^\n]\n",Questions[Question\_Number].Questions, Questions[Question\_Number].Answers) == 2)

{

Question\_Number++;

}

fclose(Database);

shuffle\_questions(Questions, Question\_Number);

///////////////////////////////////////////////Part 7//////////////////////////////////////////////

printf("Do you want to read the story ? Y/N\n");

scanf(" %c",&read\_or\_not);

if(read\_or\_not == 'Y' || read\_or\_not == 'y')

{

printf("You were a knowledgeable scholar; you had answers for questions that no one even thought of.\nBut as your knowledge circle widened, so did your enemies' numbers.\nOne of them being\nThe Obsidian Raven!\nHe hunts in the shadows, so when you were expanding your knowledge late at night in the library while everyone was sleeping,\nhe took the chance and struck you.\nWith his powerful curse, he was able to steal all your knowledge.\nThe knowledge he could not steal was who you were.\nNow, the only way to get your knowledge back is to earn it from him again by answering all of his questions, which he will ask using all of the knowledge he stole.\nGood luck, traveler :)\n");

sleep(35);

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

if(read\_or\_not == 'N' || read\_or\_not == 'n')

{

int c;

while ((c = getchar()) != '\n' && c != EOF);

}

for(int i = 0; i < Question\_Number; i++)

{

time\_t start\_time = time(NULL);

printf("%s\n",Questions[i].Questions);

while (difftime(time(NULL), start\_time) <= 60)

{

printf("\rYou have %d seconds left", 60 - (int)(difftime(time(NULL), start\_time)));

fflush(stdout);

if(\_kbhit())

{

printf("\n");

fgets(user\_answer, sizeof(user\_answer), stdin);

answered = true;

break;

}

Sleep(100);

}

printf("\n");

if(answered == false)

{

printf("Too Slow, FOCUS\n");

continue;

}

///////////////////////////////////////////////Part 8///////////////////////////////////////////////

if( user\_answer == NULL)

{

printf("Unknown Error\n");

return 1;

}

else

{

for(int i = 0; i < strlen(user\_answer); i++)

{

user\_answer[i] = tolower(user\_answer[i]);

}

user\_answer[strcspn(user\_answer, "\n")] = '\0';

if(strcmp(user\_answer, program\_ender) == 0)

{

printf("Are you sure you can face the Consequences Y/N\n");

scanf(" %c",&Try\_Again\_2);

getchar();

if(Try\_Again\_2 == 'N' || Try\_Again\_2 == 'n')

{

break;

}

else if(Try\_Again\_2 == 'Y' || Try\_Again\_2 == 'y')

{

Destiny(user\_score, Question\_Number);

sleep(5);

exit(0);

}

}

////////////////////////////////////////////Part 9//////////////////////////////////////////////////

else

{

if(strcmp(user\_answer, Questions[i].Answers) == 0)

{

printf("NICE HIT >:D\n");

user\_score++;

}

else

{

printf("OUCH >:O\n");

}

}

}

}

/////////////////////////////////////////////Part 10///////////////////////////////////////////////

printf("Do you think you are ready for him this time Y/N ?\n");

scanf(" %c",&Try\_Again);

if(Try\_Again == 'N')

{

printf("Don't Worry, We will get him next time travler\n");

sleep(5);

}

getchar();

} while(Try\_Again == 'Y' || Try\_Again == 'y');

return 0;

}

/////////////////////////////////////////////End///////////////////////////////////////////////////