**Programming in C**

**Course Project**

**Topic – Gaming Arena**

**Group No – 5**

**Group Members –**

Aditya Sodani - 3

Aishwarya Shahane - 4

Akash Shirude - 6

Shaunak Khatavkar - 28

Kirti Alkari - 30

**ABSTRACT**

This Project is prepared by Group-05 of class FYMCA-2020 batch at Vishwakarma Institute of Technology, Pune under the guidance of Prof. Dr. P P Ghadekar. In this project, we have designed  a  gaming arena(C-Arcade)  in C Programming Language.

This course project  is a console application. In the gaming arena we have Taken 4 games. To  help you to understand this course project better , There are lots of comments added within the source code. This project is aimed to teach you ‘How learning can be made fun with implementation of such topics!’

**INTRODUCTION**

We have designed a Gaming Arena for our course project which consists of 4 different games like Monty Hall, 21 number magic trick, Word search game and Quiz.

The Monty Hall game is a brain teaser based on probability. Monty Hall had a game show in which contestants were allowed to choose one of three doors. Behind each door was a prize but one prize was very good and the other two were not so good. Player had to make the best decision to get the prize.

Word search game is the game that consists of a grid of letters in which words are hidden. Player has to guess the words within the given attempts.

21 number magic trick is based on a card magic trick. 21 numbers will be shown to user and will be asked to pick one number and remember it. The program will then ask some questions to the user and then give the number that user had picked.

In quiz game, the user will be asked questions and the result of each question whether it is correct or wrong with the updated score will be shown. The goal is to get the maximum score.

All these game programs were developed using C language.

**OBJECTIVES**

1. To design and develop games in C-language
2. To have multiple games under one window
3. To understand concept of Monty Hall problem: a probability puzzle.
4. To understand the working of 21 cards magic trick with the help of 21 number magic trick program.
5. Make use of C-language to make innovative games.

**APPLICATIONS:**

* Game 'Monty -hall problem' is a probability based puzzle. It helps us to understand concepts in probability in a unique way.
* Game words search can be used to test knowledge regarding C language.
* 21 number magic is an implementation of 21 card magic trick, we can analyse the working of this program and apply it in real life
* Game quiz is designed to enhance basic computer knowledge , general knowledge and to improve your English verbal communication.

**ALGORITHMS & FLOWCHARTS**

Each game has its separate program

system(“filename.exe”) function is used to call other multiple programs from the game arena program.

1. **Monty Hall Problem**

**Description**

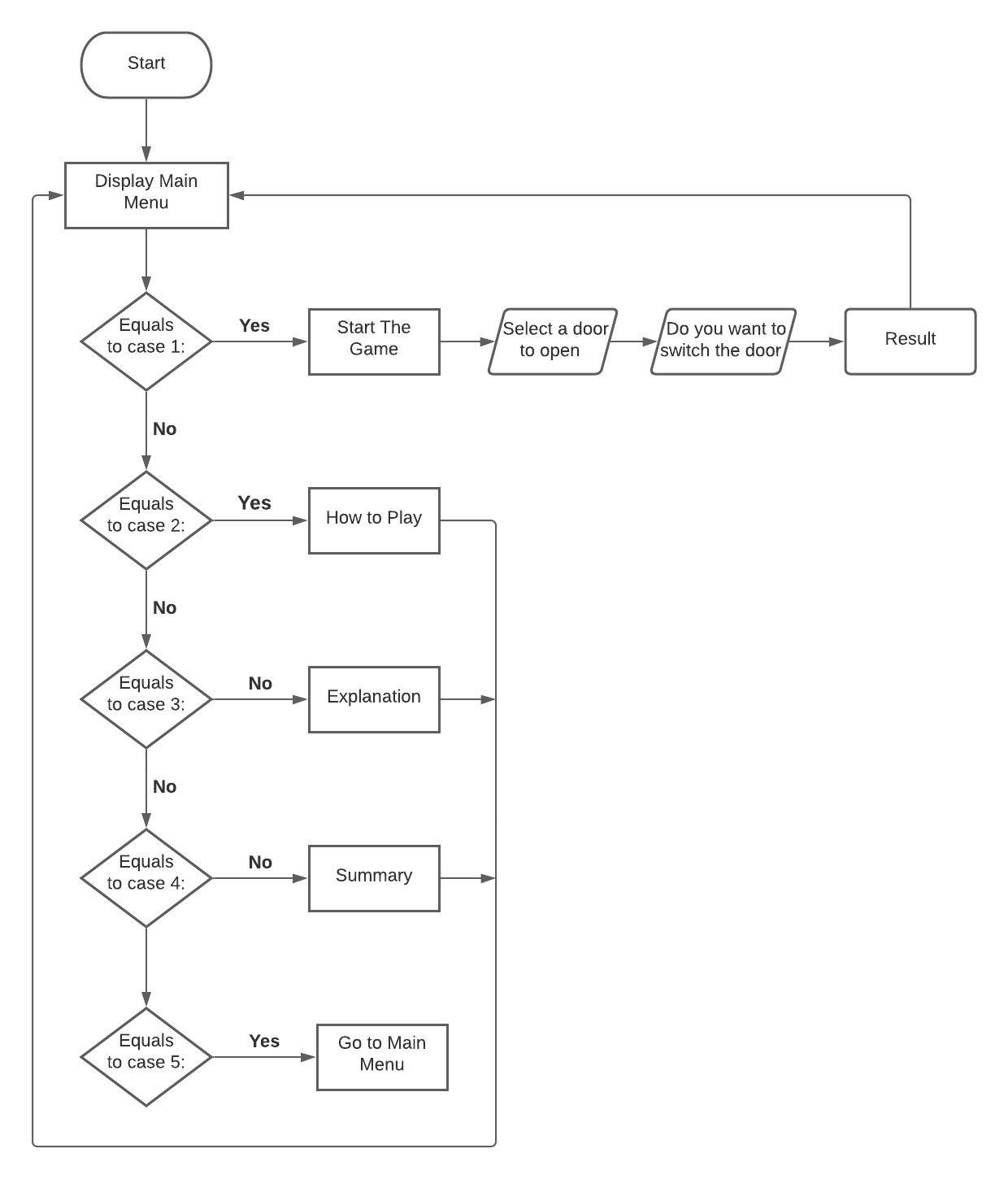
The Monty Hall problem is a counter-intuitive statistics puzzle:

* There are 3 doors, behind which are two goats and a car.
* You pick a door (call it door A). You’re hoping for the car of course.
* Monty Hall, the game show host, examines the other doors (B & C) and opens one with a goat. (If both doors have goats, he picks randomly.)

Here’s the game: Do you stick with door A (original guess) or switch to the unopened door? Does it matter?

Surprisingly, the odds aren’t 50-50. If you switch doors you’ll win 2/3 of the time!

**Flowchart :**



**\*\*\* Detailed Algorithm of Code \*\*\***

* An array is created to store the values for 3 doors doors[3]. Here value 1 will represent a “Car” while 0 represents “Goat”. Hence there will be two 0s and one 1 in the array.
* Now the 1 (Car) should be assigned to random index of the array from index[0] to index [2].
* Srand() and rand() functions are used to generate random variable from 0 to 2
* This is to assign 1 to random index from 0 to 2
* Then at this random index, 1 will be assigned.
* A separate variable “carDoor” will store the number of the door that has “car’ behind it, That door number is the randomly generated index + 1 (as arrays start from 0)
* Now in array door[3] we can have 3 sets of values {1,0,0} or {0,1,0} or {0,0,1}
* 3 doors will be printed using drawDoors(int) function. The parameter passed to this function will be the number of doors opened. Initially no door is opened hence we will pass drawDoors(0).
* Now player is asked to open any one of the 3 doors, he/she has to enter any number from 1 to 3. This input will be stored in a variable “choice”. Choice variable is used many times throughout the program
* Now a for loop is started from I =0 to I < 3 (3 = number of doors)
* **(Inside for loop)**
* **(Select a door which does not have a car behind it and is not the door selected by player)\**

**This will be the door that will be opened and shown to the player**

**If ( door[i] ==0 && choice-1 != i)**

This means when door[] value at index I is not equal to 0 (when there is a goat behind the door) and Choice - 1 value is not equal the iterator I (When the door no. picked by user is not equal to the door were looking at right now)

* (**inside if block)**
* Now a door which has Goat behind it will be shown to the player using drawDoors(1) function. 1 door is opened hence drawDoors(1) is passed
* Player now knows that this particular door definitely doesn’t have a CAR behind it
* **(Break out of if and for Loop)**
* Now there are 3 doors out of which 1 is chosen by the player to open and 1 door program has opened and shown to the player. So now only 1 door in the set is left.
* We will assign this door no. to a variable called “optionDoor”
* A for loop from 0 to 3 is used to assign this value where condition is if(iter != choice-1 && iter!=i) Here **i** is the iterator value from previous loop which has the no. of exposed door
* This optionDoor is the door that will now the player can decide to switch to from initial “choice” door
* A function switchDoor() is used to perform actions from now on
* Now player is asked if he/she wants to switch to the other “optionDoor” or stick to the original “choice” door.
* Player will answer in either “Y/y” (Yes) or “N/n”(No)
* **If player decides to switch and answers (YES)**
  + **Check if the optionDoor has a CAR behind it or a GOAT behind it**
  + **IF (door[optionDoor – 1] == 1) (1= Car)**
  + Then print that player has won a car
  + **ELSE** -> Print that Player has lost the game
  + The doors will all be opened and displayed to player and a RECAP will be shown which will show the previous choice of user and if he/she decided to switch or not.
  + storeRecord(char,int) function will be called to store the winning/losing records in a file
* **If player decides NOT to switch and answers (NO)**
  + **Check if the “choice” has a CAR behind it or a GOAT behind it**
  + **IF (door[choice – 1] == 1**) (1= Car)
  + Then print that player has won a car
  + **ELSE** -> Print that Player has lost the game
  + The doors will all be opened and displayed to player and a RECAP will be shown which will show the previous choice of user and if he/she decided to switch or not.
  + storeRecord(char,int) function will be called to store the winning/losing records in a file
* Game logic ends here
* **Other functions that were used in the program**

1. **Char menu()**

* This function is used to print the menu window
* This will have the following
* 1. Start the game

2. How to play

3. Game Explanation

4. Summary of previous contestants

5. Back to Main Menu

* Then player will be asked to enter any option of his/her choice from 1-5
* This value will be accepted in character to avoid any runtime errors
* Character value that player will enter is returned

1. **Void howToPlay()**

* This function will print the details will explain the flow of the game.

1. **Void montyHallExplanation()**

* This function will print the texts explaining the monty hall problem

1. **Void summaryTable()**

* This function will print the summary of previous winning records of contestants
* This will be printed in a tabular format like following

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of players | Winners | Percent Winners |
| Switched | Integer | Integer | float |
| Didn’t Switch | Integer | Integer | float |

* All this data will be fetched from a function readRecord();

1. **Void readRecord()**

* This function will read the file of winning records
* A file “scoreRecord.txt” will store the records of winners (switched/not switched) players.
* This function will read those values and store it in variables within the program
* 6 variables are there to store the records
* 1) Number of players who decided to switch

2) Number of players who won after switching

3) Percentage of players who won after switching

4) Number of players who decided to not switch

5) Number of players who won by not switching

6) Percentage of players who won without switching

1. **Void storeRecord(char sw,int win)**

* This function will update the winning and player records in the “scoreRecord.txt”
* Here the character parameter will be passed with value of either ‘s’ or ‘d’ where ‘s’ means switched and ‘d’ means didn’t switch
* And the integer parameter that is passed will have either 0 or 1. 0 for losing and 1 for winning
* With the help of these 2 values we can update values at appropriate places in the file.
* Following are the changes that will be made in the file
* **(Percentage will be calculated for every condition)**
* **(All the 6 elements will be rewritten in the file only some will change for certain conditions)**
* 1) When sw == ‘s’ and win == 1 (when player switches and wins)
  + Only increment the no. of switch players and no. of players won by switching

2) When sw == ‘s’ and win == 0 (when players switches and loses)

* Only increment the no. of switch players

3) When sw == ‘d’ and win == 1 (when player does not switch and wins)

* + Only increment the no. of not switch players and no. of players won by not switching

4) When sw == ‘d’ and win == 0 (when player does not switche and loses)

* Only increment the no. of not switch players

1. **Void drawDoors(int open)**

* This function will print the 3 doors
* Parameter open will have either 0,1 or 2 as a parameter each value describing the number of doors open. For example 0 for zero doors opened and so on
* A variable called “doorNo” will store the number of the door that is to be opened when open value is 1
* When function start, doorNo will hold the value of i from the for loop in main function
* When open is equal to 2 means when both doors are to be opened. In reality all three will be opened
* carDoor variable will tell us which door has Car behind it and doors will be printed with this information.

1. **Void winLose(bool win)**

* This function will print if player loses or wins
* Boolean parameter will either have ‘true’ or ‘false’ as a parameter
* True means the player has won and appropriate message will be printed
* False means the player has lost and appropriate message will be printed
* **Two functions that are used to add a delaying feature while printing**

**1)** **void delay(int milliseconds)**

* This function will add a delay of milliseconds passed through the parameter

**2) void printDelay(char str1[], int delayTime)**

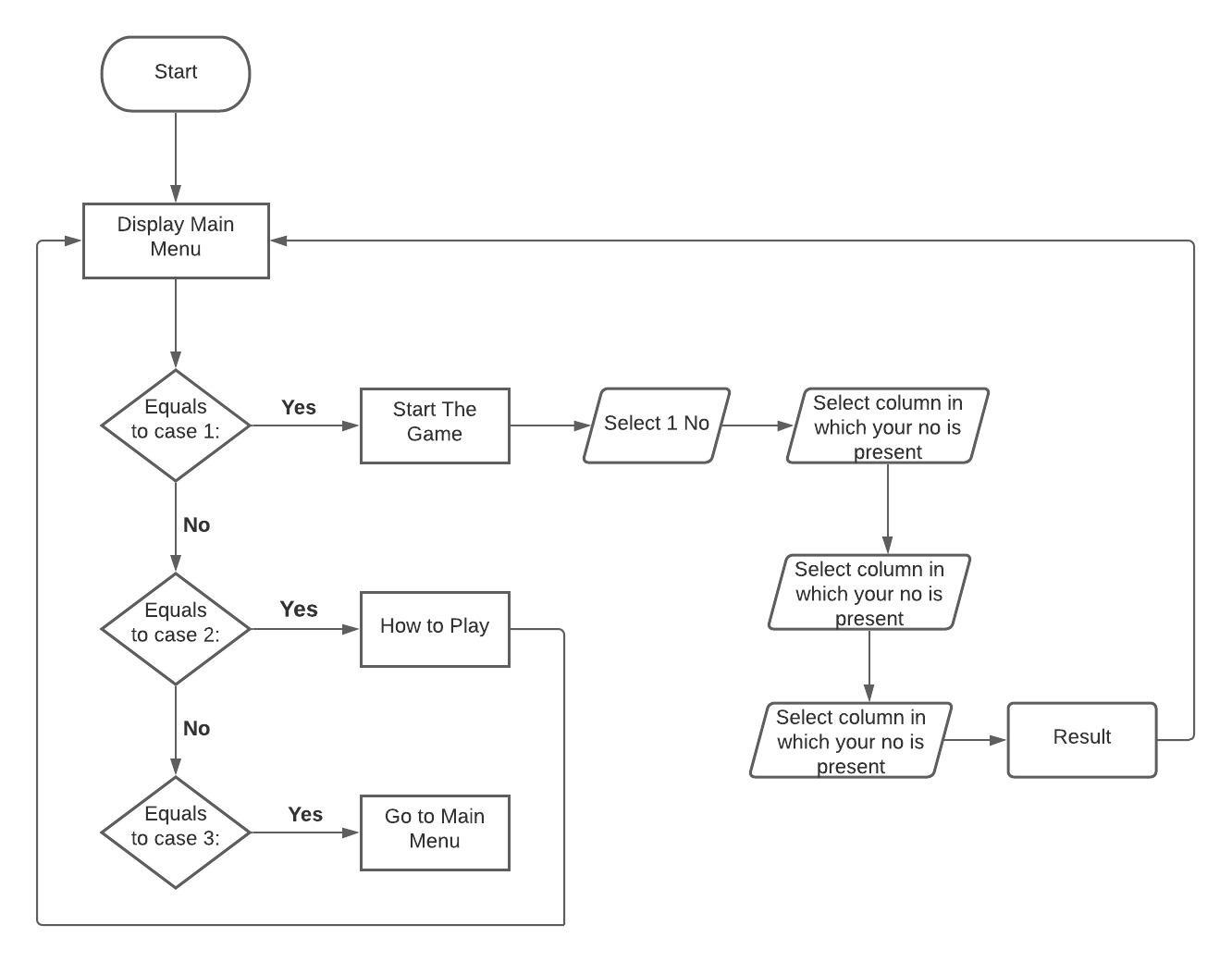
* The string passed in parameter will be the string to be printed.
* There should be a delay after every character hence
* A loop is created from 0th index to end of the string and print each character and call delay() function with appropriate delay time.’

1. **21 Number Magic Trick**

**Description**

* This game is based on 21 card magic trick
* Any 21 cards are picked from a pack of cards.
* Then spectator is told to pick any one of the 21 cards and remember it.
* Now, the cardician will shuffle the cards
* Then he/she will start dealing the cards in 3 sets and ask the spectator to play close attention to each card. After dealing all the cards the spectator will tell the cardician about in which set his/her card was.
* Now the cardician will recollect the sets with cards faced down, but the set that spectator’s card was in will be kept in the middle of the other 2.
* This will be repeated 2 more times. 3 times in total.
* The card that spectator selected will be the 11th card in starting from either side of the 21 card pack. In other words, it will be the middle card

**Flowchart**



**\*\*\* Detailed Algorithm of Code \*\*\***

1. In this program, numbers are used instead of cards. Rest of the process remains the same.
2. 21 random numbers from 1 to 50 will be generated.
3. An array cards[21] is used to store 21 unique numbers. The numbers should not be repeated, a function generateRandom() is used to generate those unique numbers without repeatation.
4. These 21 numbers are shown to the player and he is asked to pick any one number and remember it
5. Now the numbers are to be shown to the player in a set of 3s
6. dealCards() function is used to achieve this. Cards[21] array will be shown to player from index 0 to 20
7. cards[0] cards[1] cards[2]

cards[3] cards[4] cards[5]

and so on… until cards[20].

1. Then the player will be asked enter the number of column where his/her number is and that column number will be stored in variable “choice” and this variable determine future operations
2. As we discussed above, 3 sets are created of 7 cards/numbers in each set. So for these 3 sets, 3 arrays are created namely card1[7], card2[7] and card3[7].
3. Now these 21 cards will be distributed to the 3 arrays card1,card2 and card3 according to the column numbers. Numbers in column 1 will be stored in card1[7] , 2 in card2[7] and 3 in card3[7]. Function sectionCards() is used to achieve this task
4. Cards are picked up with faced down in other words sequence of cards is reversed so similarly the 3 columns/sets of numbers have to be reversed before recollecting.
5. Arrays cards1[7],cards2[7] and cards3[7] are reversed.
6. The column where player’s number is present must be put in between the rest of the 2 columns/sets. But if it is already in the middle column/set then no need to rearrange.
7. If the “choice” variable is 1 then cards1[] will be swapped with cards2[]
8. If the “choice” variable is 3 then cards3[] will be swapped with cards2[]
9. This way the column with player’s number will be put in the middle
10. Now the 3 arrays which are now reversed and rearranged will be stored in the cards[21] array. sectionCardsReverse() function is used to do this. Numbers will be stored in reverse order in the cards[21] array means cards1[7],cards2[7],cards3[7] arrays will be stored in reverse order. Cards1[7] array will be stored in index 13-20 of cards[21], cards2[7] in index 7-13 of cards[21] amd cards3[7] in index 0-6 of cards[21]
11. Now Repeat from Step (4) 3 times in total
12. After repeating process from Step (4) to Step (18) total 3 times the player’s number will be the 11th or the middle number of the cards[21] array
13. Middle element of cards[21] will be the 11th element or the element at 10th index(cards[10])
14. Now player will be shown the number at 10th index of cards[21] array (cards[10]), the number he/she had picked earlier
15. The game ends here

* Functions used in the game

1. **Char menu()**

* This function is used to print the menu window
* This will have the following
* 1. Start the game

2. How to play

3. Back to Main Menu

* Then player will be asked to enter any option of his/her choice from 1-3
* This value will be accepted in character to avoid any runtime errors
* Character value that player will enter is returned

1. **Void howToPlay()**

* This function will print details on how to play the game

1. **Void generateRandom()**

* In this function 21 unique random numbers will be generated and stored into the array cards[21]
* A while loop is created starting from count = 0 until when count <21
* A random variable will be generated from 1 to 50. Now we have to make sure that no number is repeated in the array
* A variable “found” is created with value 0
* A for loop is started from i=0 to i<21
* Now this random number will be compared with the array element at ith index (cards[i] == randomnumber)
* If the number is matched then we assign found = 1 and break out of the loop
* Now if found ==0 then that means the random generated number is not present in the array already
* Then the random number will be assigned in the array at count(th) position
* Repeat from while loop until loop is over
* Random 21 numbers are allotted in the array cards[21] now

1. **Int deal\_cards()**

* In this function the numbers are showed to the player in 3 columns/sets
* Now these numbers should get printed in a way where 3 numbers are printed in a line and so on
* For this purpose a nested for loop is used
* Outer loop goes from 0 to 7 and inner loop goes from 0 to 3.
* After printing player will be asked in which column is his/her number then that “choice” will be returned back

1. **Void sectionCards()**

* In this function the elements from cards[21] will be distributed in 3 arrays cards1[], cards2[] and cards3[]
* Now these elements are stored in a column wise way
* For this 3 for loops are created
* **Column1** **Column2** **Column3**

Cards1[0] = cards[0] cards2[0] = cards[1] cards3[0] = cards[2]

Cards1[1] = cards[3] cards2[1] = cards[4] cards3[1] = cards[5]

.

.

.

Cards1[6] = cards[18] cards2[6] = cards[19] cards3[6] = cards[20]

1. **Void sectionCardsReverse()**

* In this function the elements from the 3 arrays will be restored in the cards[21] array.
* Now the elements will be stored in reverse order. ‘
* 7 elements from cards1[7] array will be stored in last 7 indexes of cards[21] array.
* **Two functions that are used to add a delaying feature while printing**

**1) void delay(int milliseconds)**

* This function will add a delay of milliseconds passed through the parameter

**2) void printDelay(char str1[], int delayTime)**

* The string passed in parameter will be the string to be printed.
* There should be a delay after every character hence
* A loop is created from 0th index to end of the string and print each character and call delay() function with appropriate delay time.’

1. **Word Search Game**

A **word search**, **word find**, **word seek**, **word sleuth** or **mystery word** puzzle is a word game that consists of the letters of words placed in a grid, which usually has a rectangular or square shape. The objective of this puzzle is to find and mark all the words hidden inside the box. The words may be placed horizontally, vertically, or diagonally.

User will get 10 attempts to get 7 words correct from the grid.

**Algorithm**:

STEP 1: START

STEP 2: Initialize string array input[][] which contains puzzle, line by line and initialize score=0 and attempt =1

STEP 3: Initialize array of string ans[][] which contains all the answers of puzzle

STEP 4: display

1. Start the game

2. How to play

3. Back to main menu

STEP 5. Read choice

STEP 6: if choice =1

6.1 print puzzle line by line by printing input[][] array using for loop

6.2 if score=7 or attempt =10 then goto STEP 6.8

6.3 enter str

6.4 if str has already entered before then display warning message and goto 6.6

6.5 if str matches to one of the string in ans[][] array then score=score+1

6.6 attempt ++

6.7 goto 6.1

6.8 print ‘score’ along with appropriate message

STEP 7: if choice=2 then display all the information related to game

STEP 8: if choice=3 then exit the game

STEP 9: END

1. **Quiz Game**

**Description**

Three quizzes are there in the program : verbal, general knowledge, computing minds. Each quiz will have 10 questions. Player has to pick one correct option from 4 options provided to him/her for each question.

**Algorithm:**

1. Start the program.

2. Initialize all variables

3 Display Game menu and accept input from user.

4. If (Case==1) 🡪 Goto Main1 Function()

4.1 Initialize all variables in Main1() Function and then enter into do while loop.

4.2 Display Quiz masters league menu.

4.3 Declare Label label1.

4.3.1 Choose a quiz subject input from user.

4.3.2 If input==4 then goto main function.

4.3.3 If 4>input<1 Show message to display a valid input and goto label1 () again.

4.4 If case=1 in Quiz masters league then start verbal quiz and display score using verbal function()

4.5 If case=2 in Quiz masters league then start Gk quiz and display score using Gk function.

4.6 If case=3 in Quiz masters league then start Computer minds quiz and display score using comp function.

4.7 If case= 4 Goto main function.

4.8 After Displaying result by press any key and goto main function.

5. If case=2 Display How to play Screen.

5.1 Press any key to go back to main menu.

6. If case =3 Go to the Game arenas main menu

7 If case= 4 Exit the game.

8. Declare getch1() and initialize all variables

9 Declare label2

9.1 If temporary variable == Getch function.

9.2 If temporary variable is A|| B||C||D||a ||b ||c ||d Return temporary variable

9.3 Else print select valid choice and goto label2.

10. Initialize verbal function

10.1 Set count==0 print the question

10.2 If function getch = correct options increment count and goto next question.

10.3 After all questions return count to the function which will display total score when the function is called.

11. Initialize Gk function

11. 1 Set count==0 print the question

11.2 If function getch = correct options increment count and goto next question.

11.3 After all questions return count to the function which will display total score when the function is called.

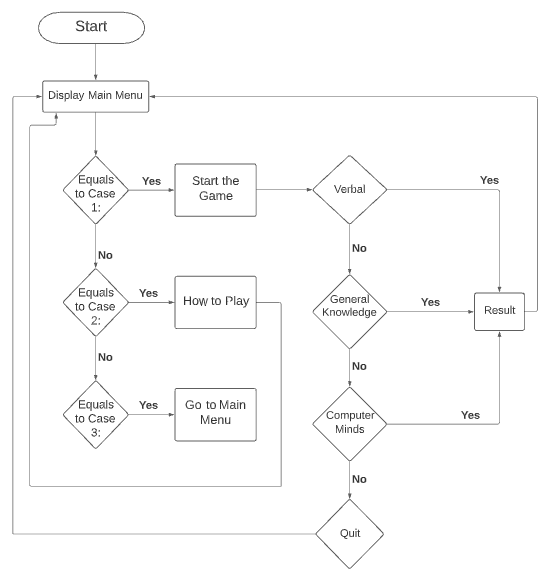
12. Initialize Comp function

12. 1 Set count==0 print the question

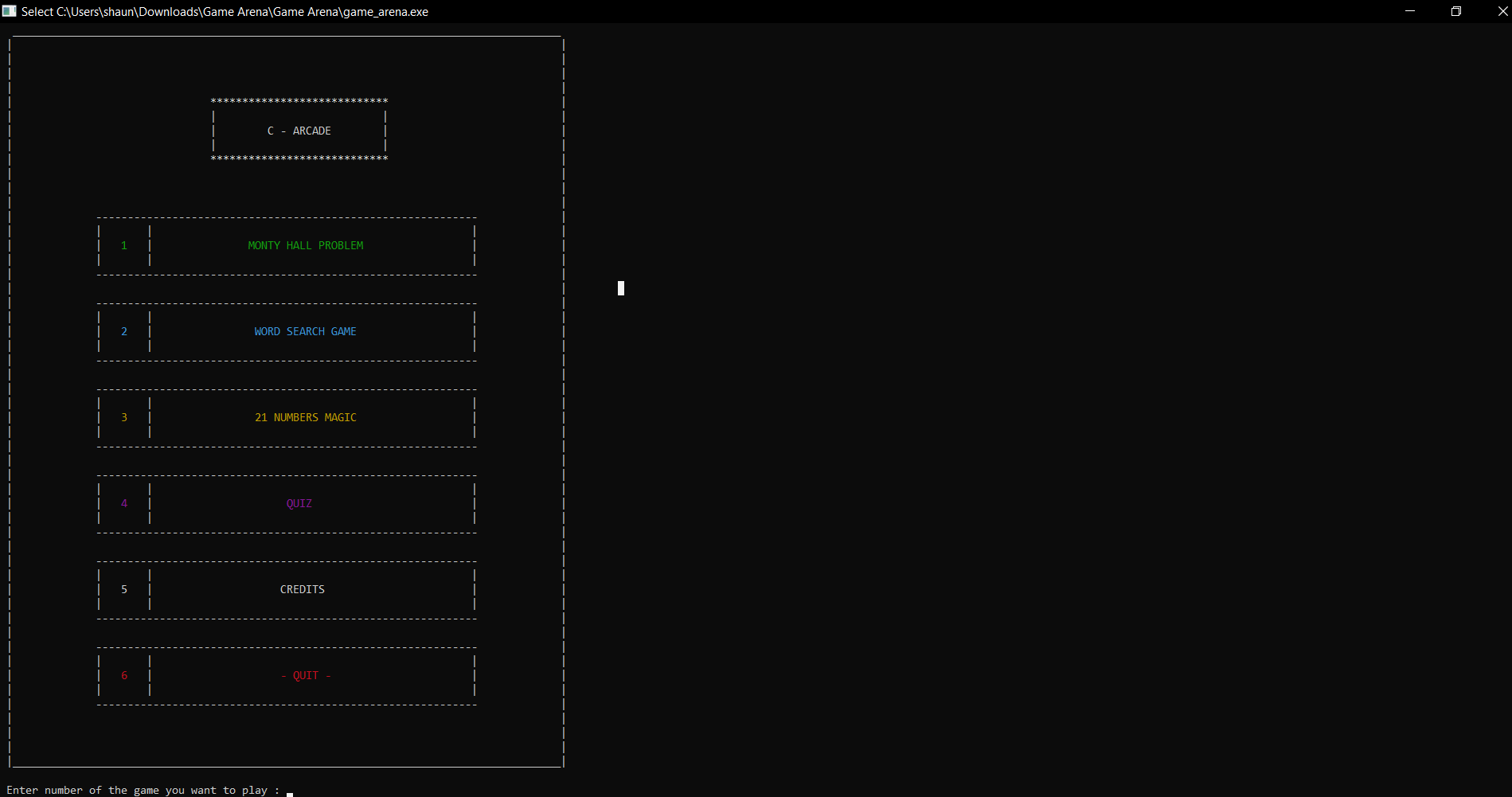
12.2 If function getch = correct options increment count and goto next question.

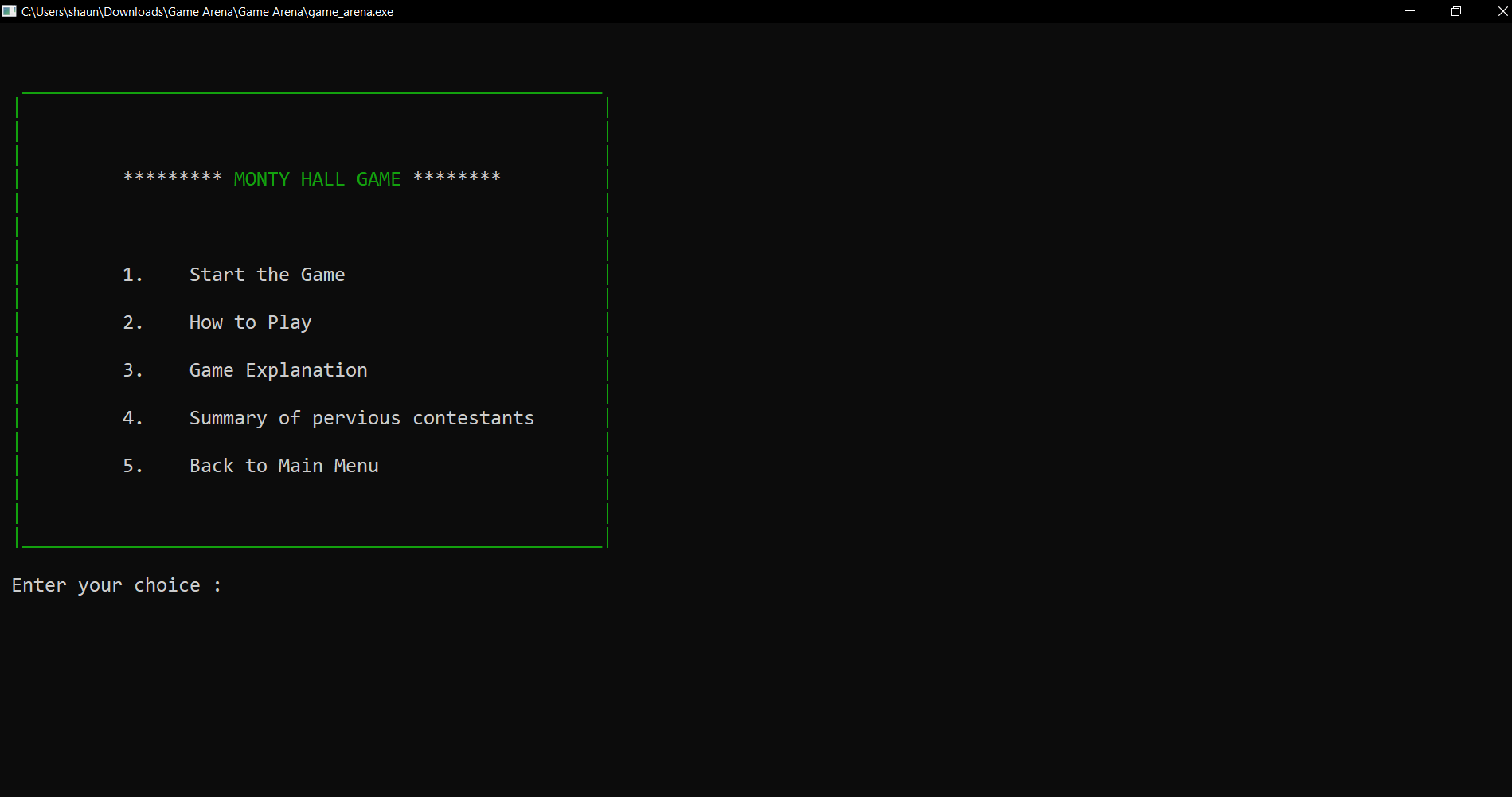
12.3 After all questions return count to the function which will display total score when the function is called.

**Flowchart :**



**OUTPUT SCREENSHOTS**





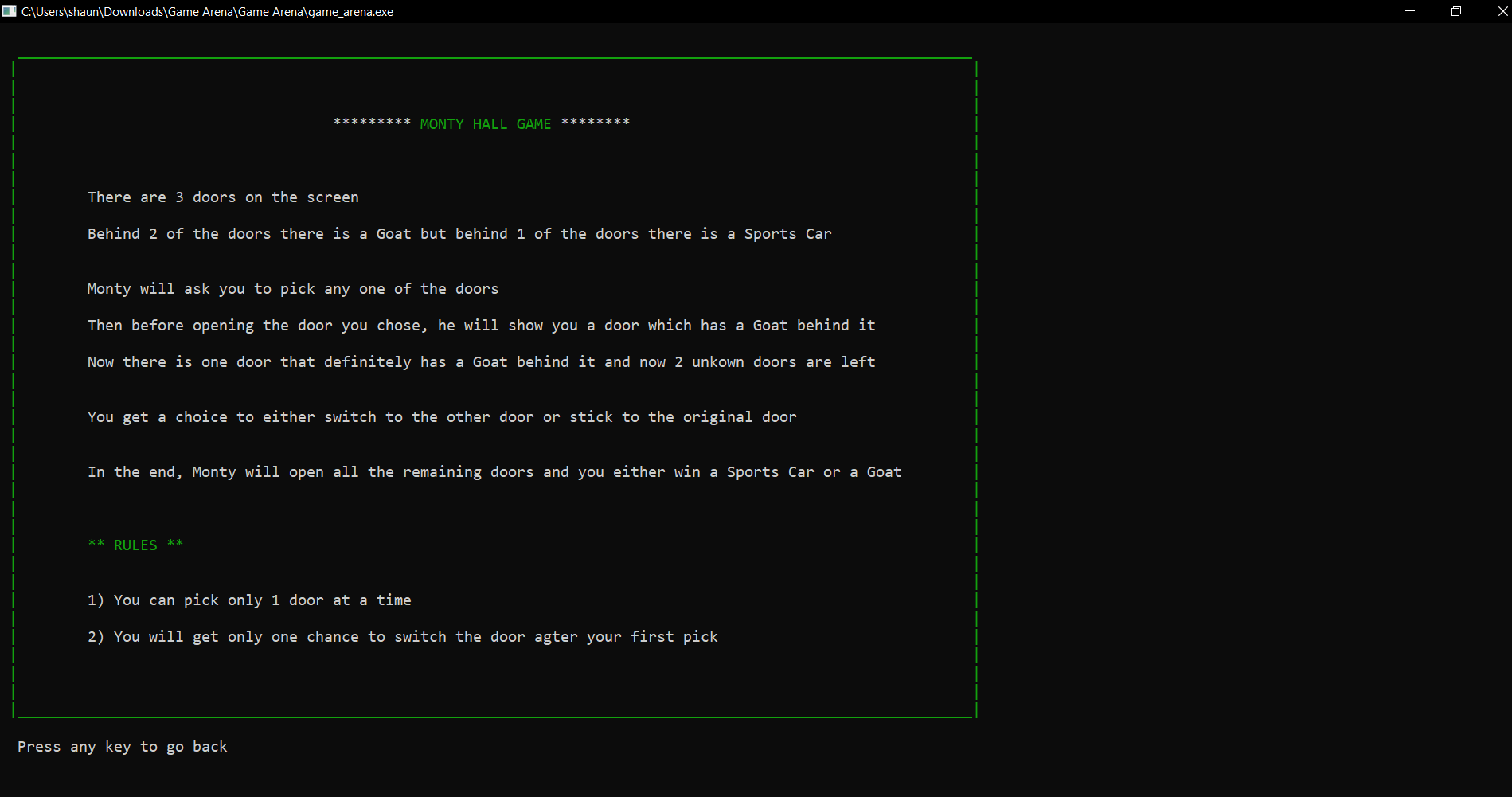


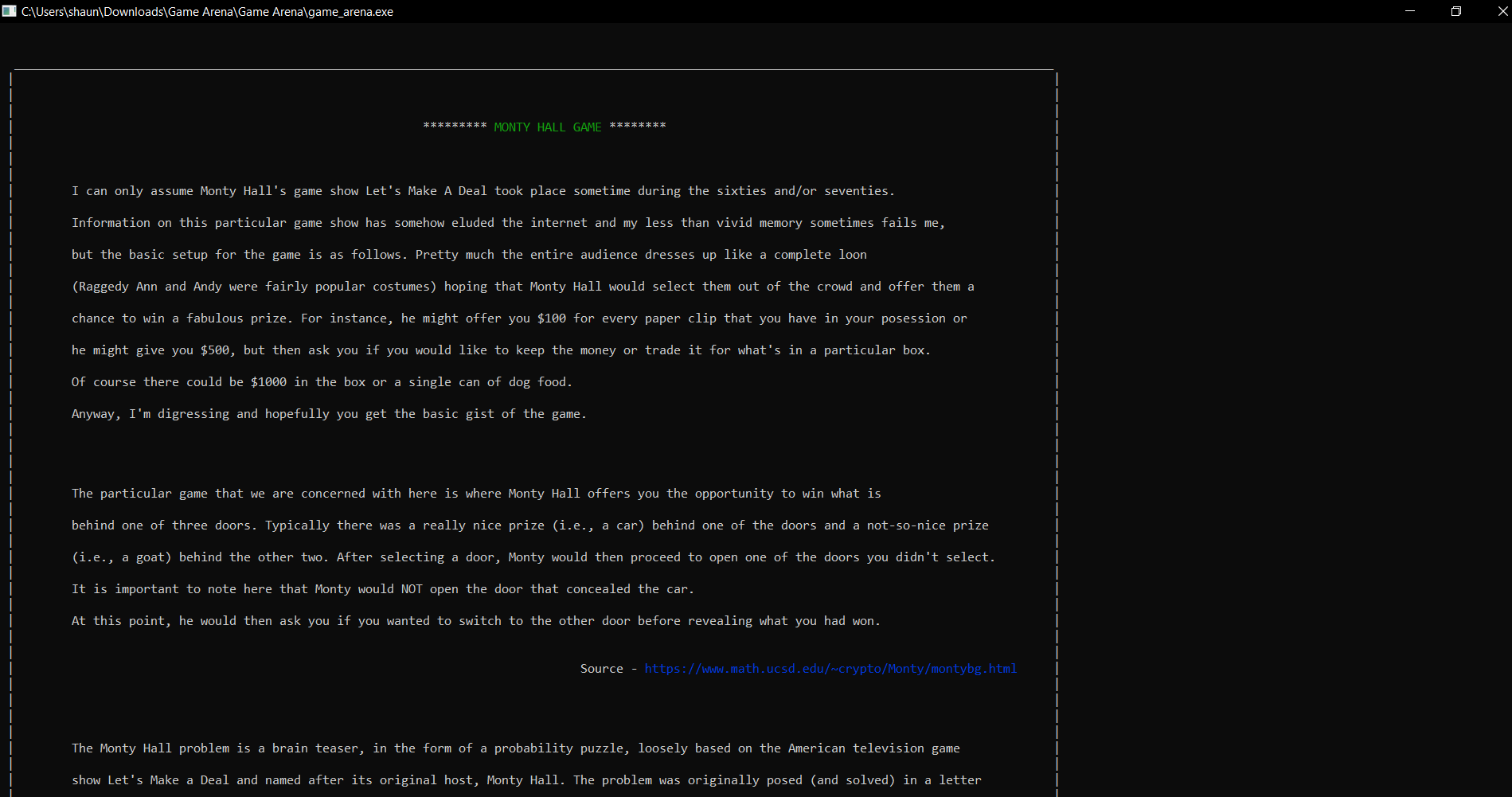


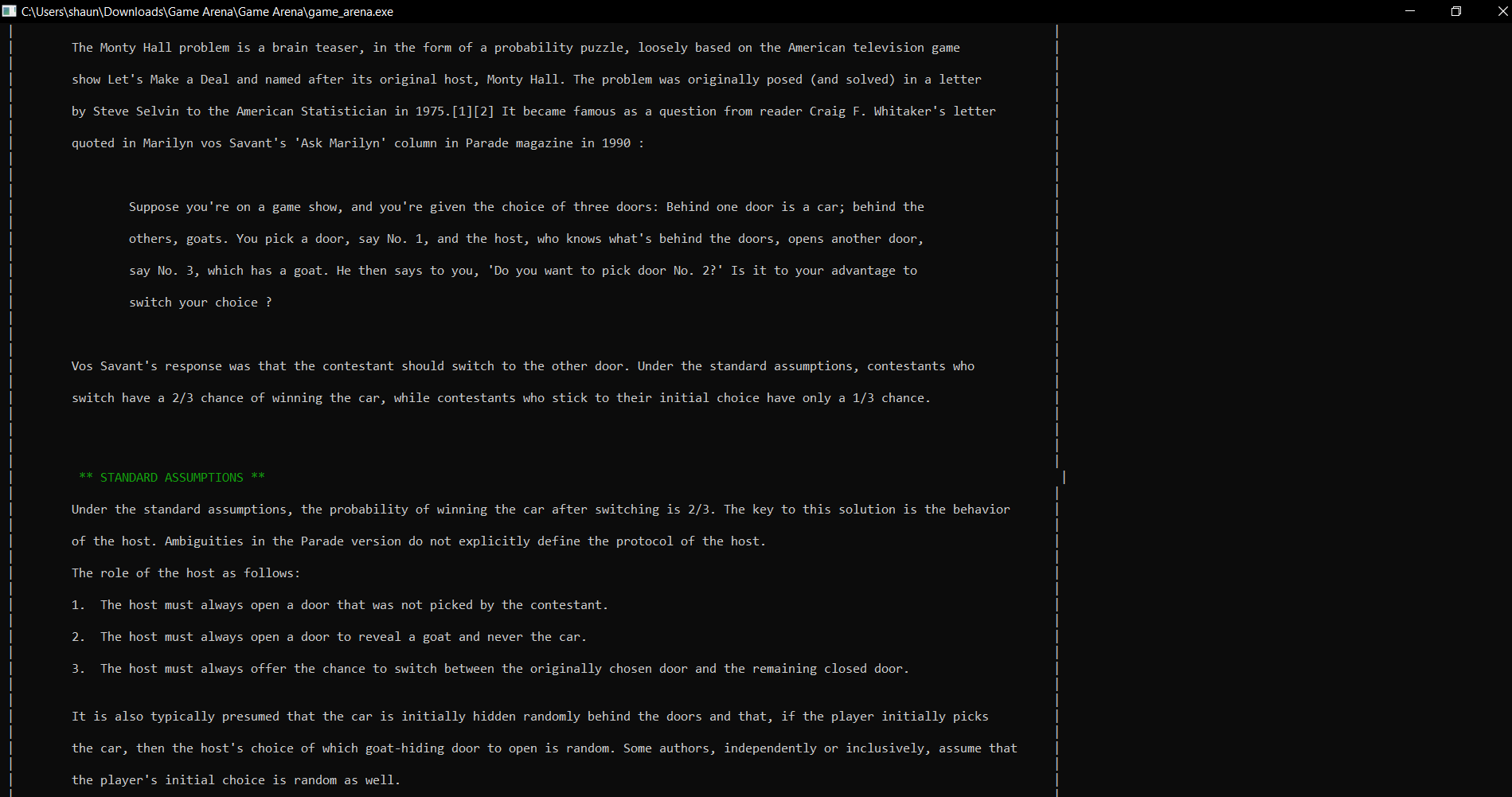


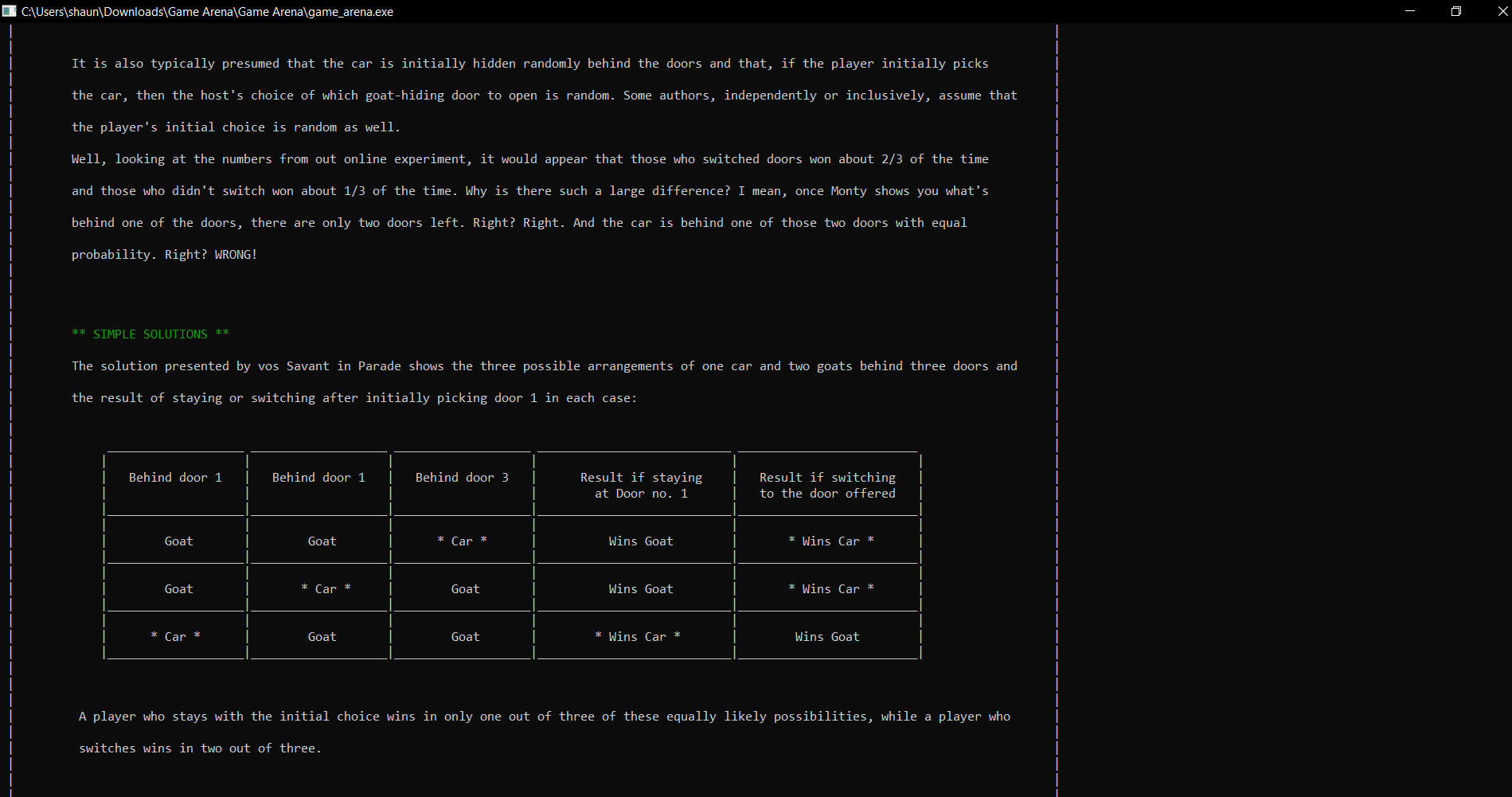


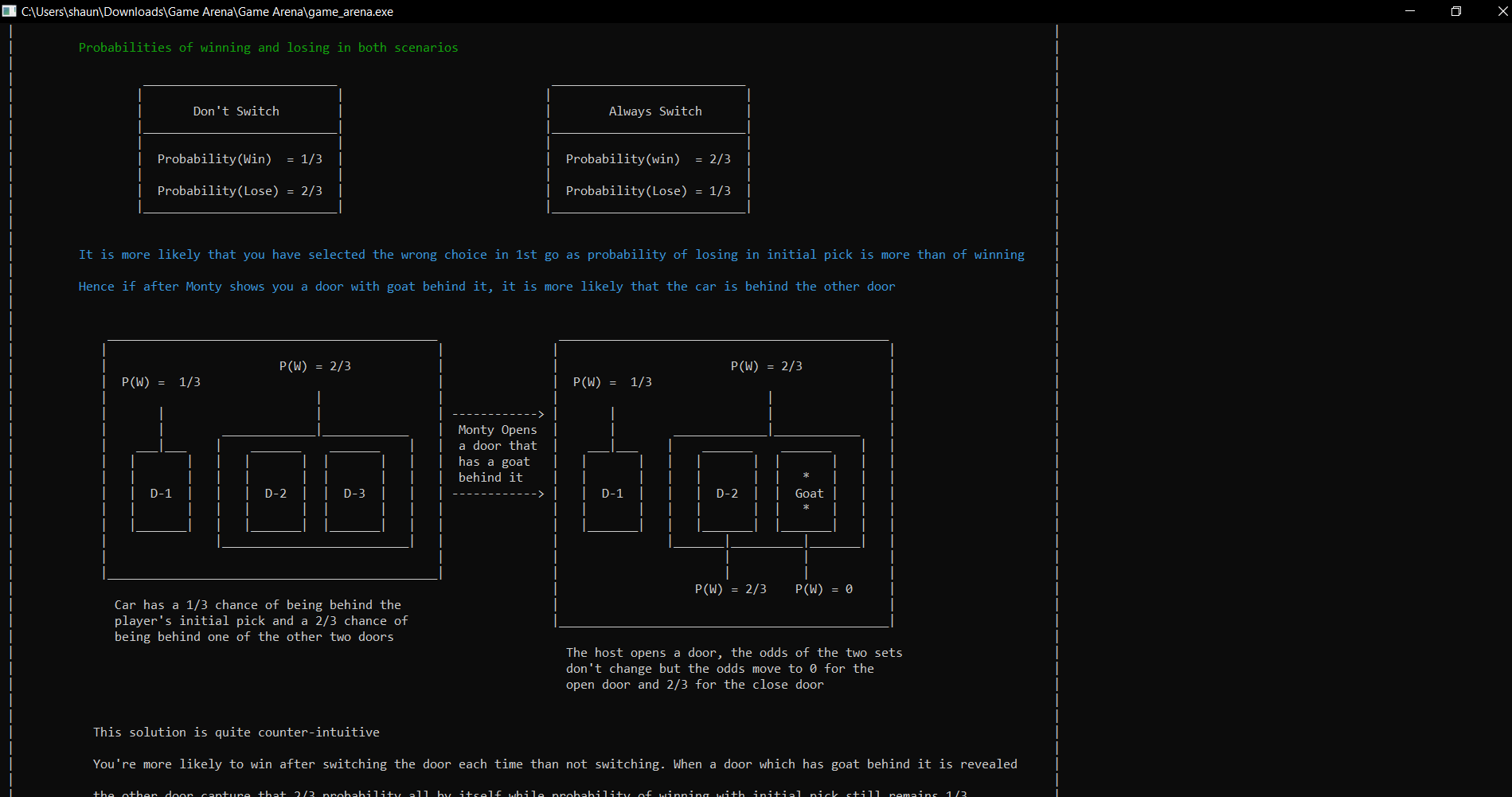


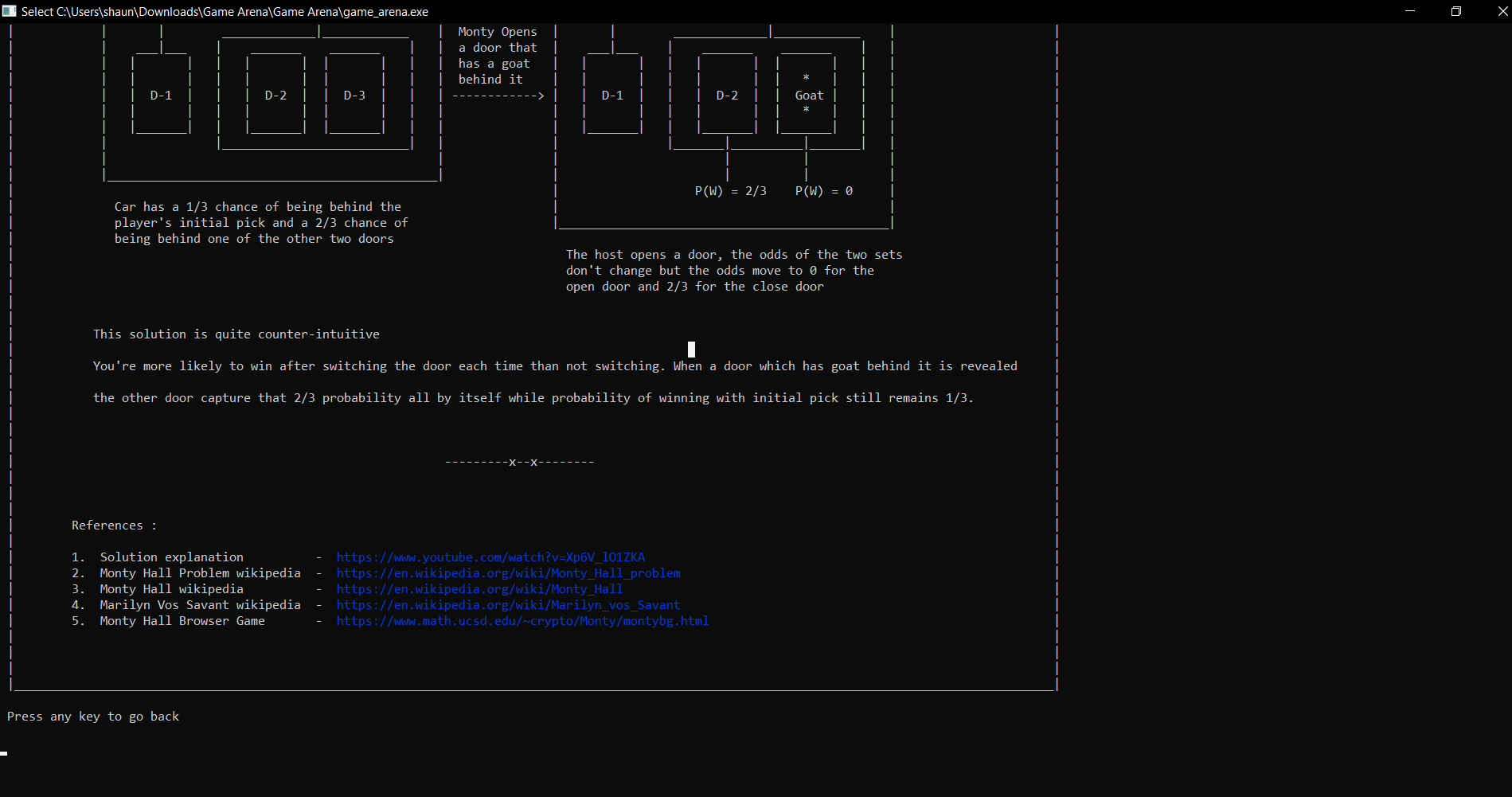


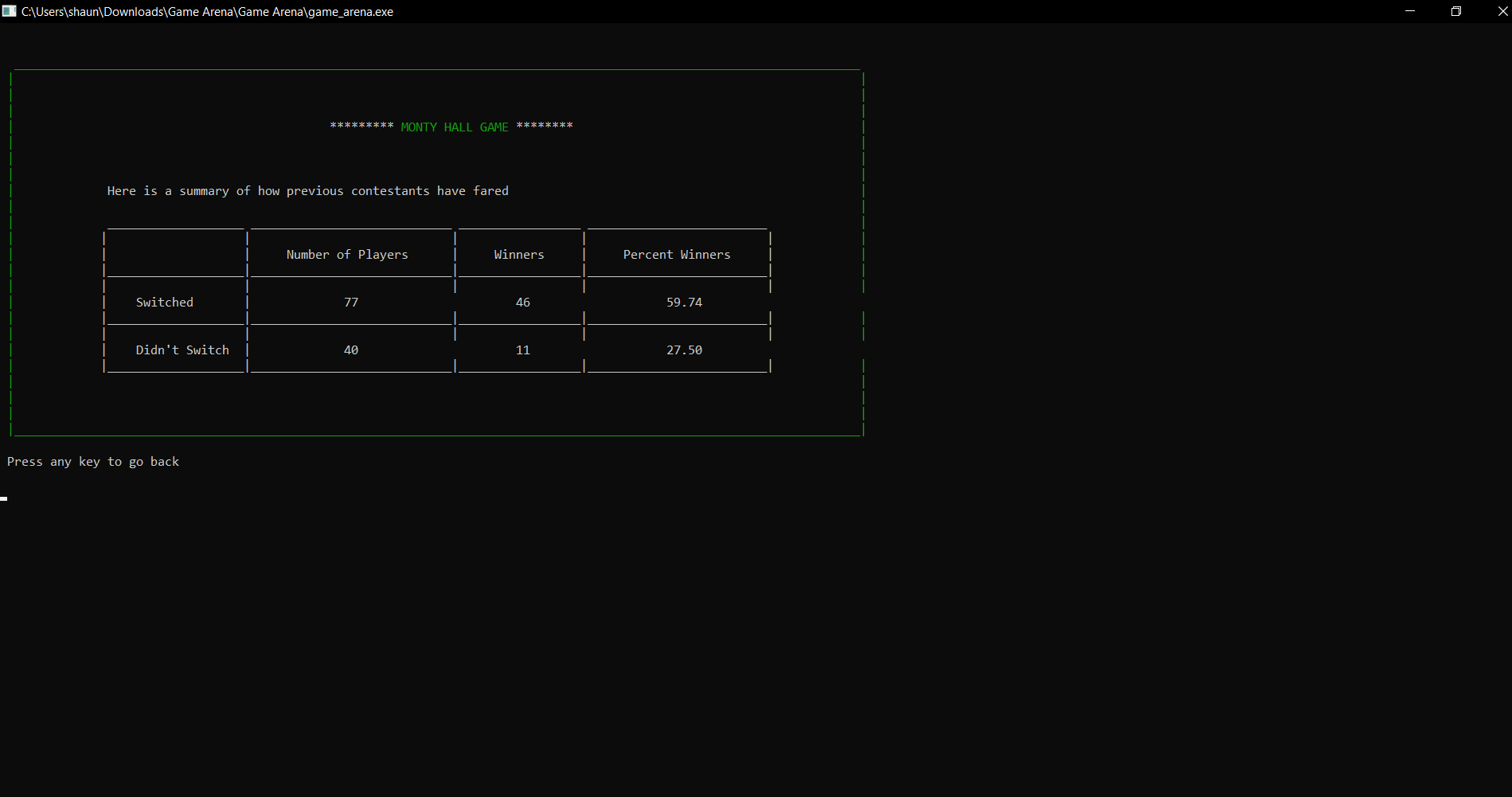


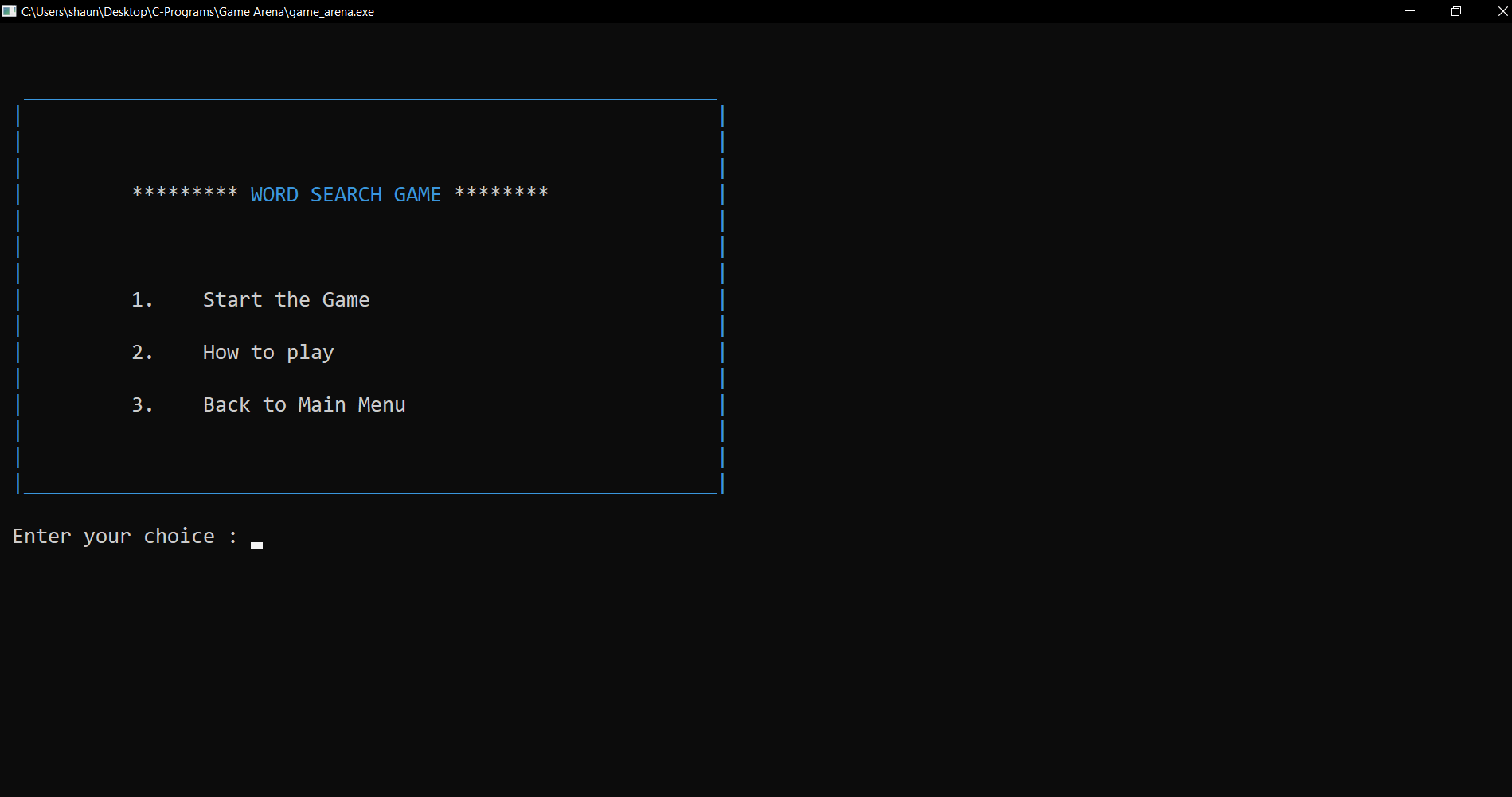


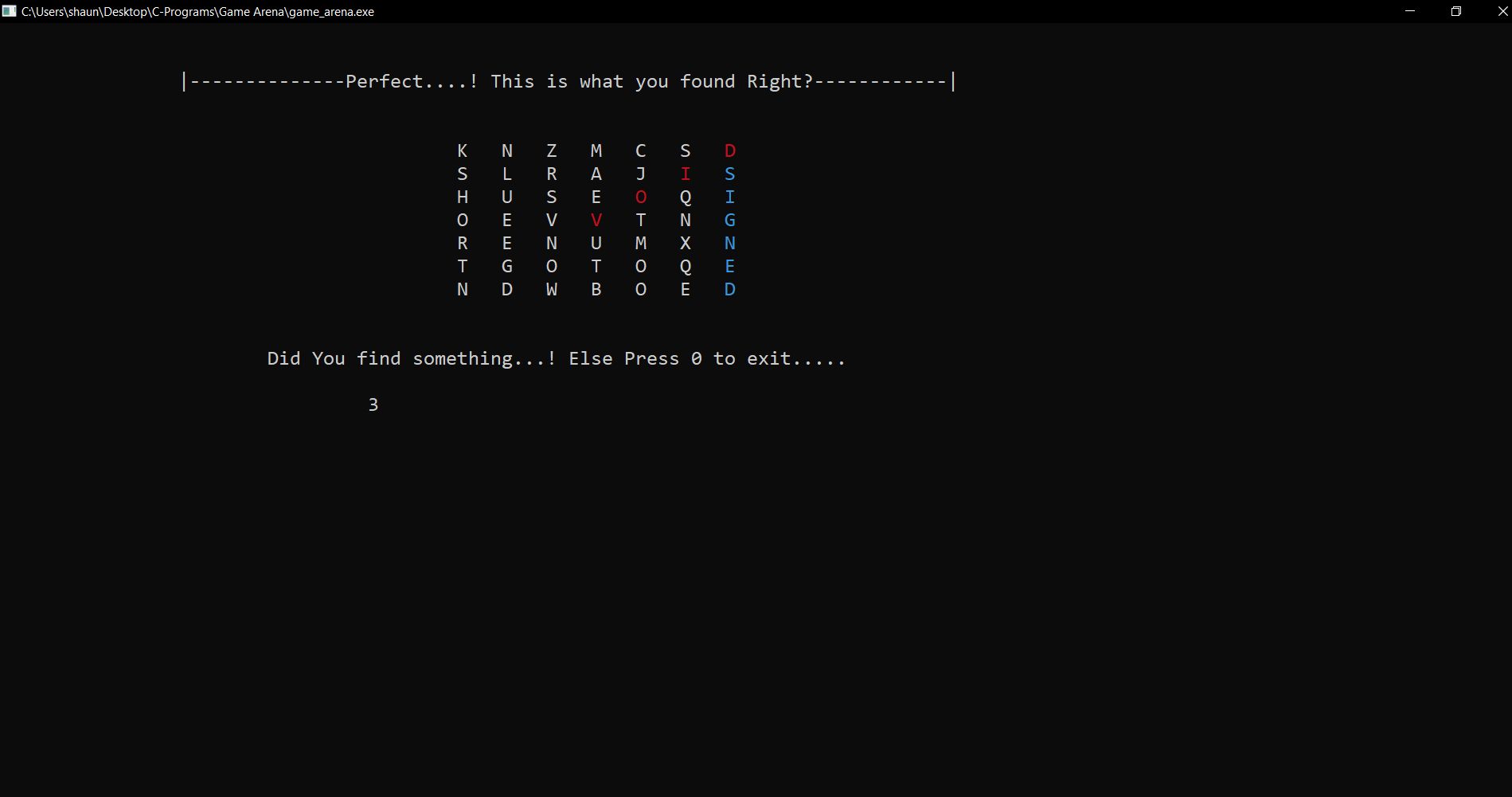


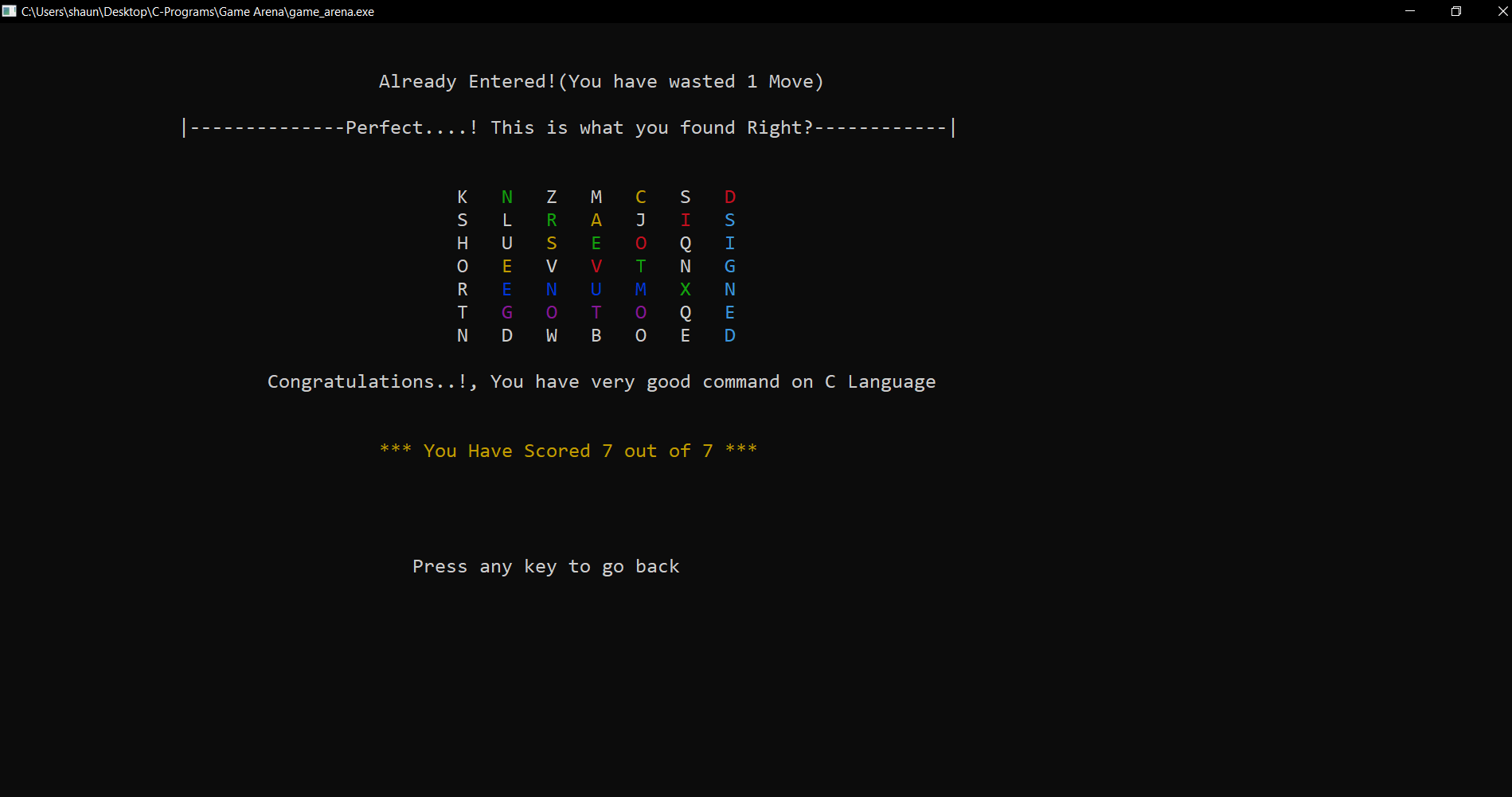


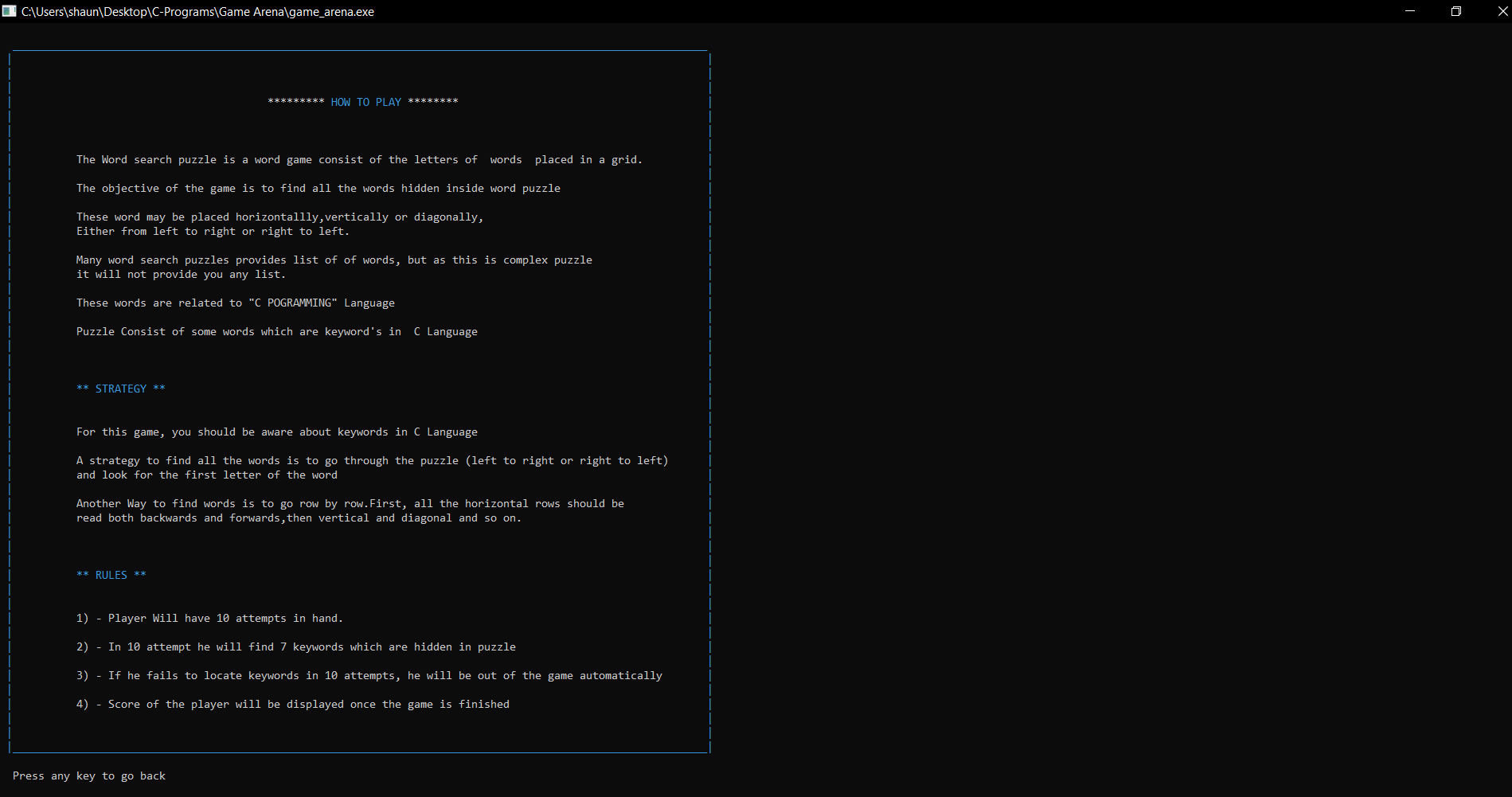


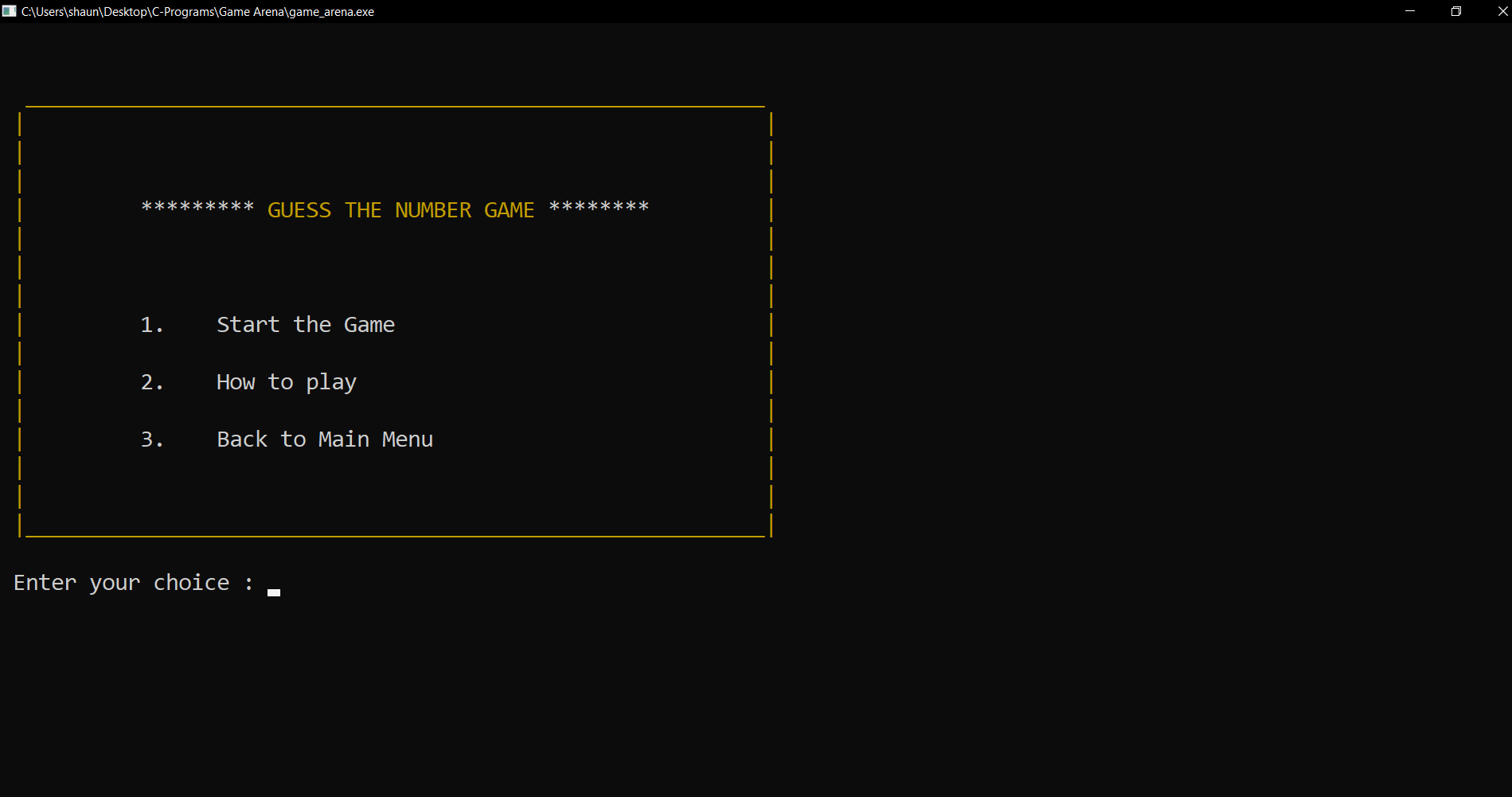


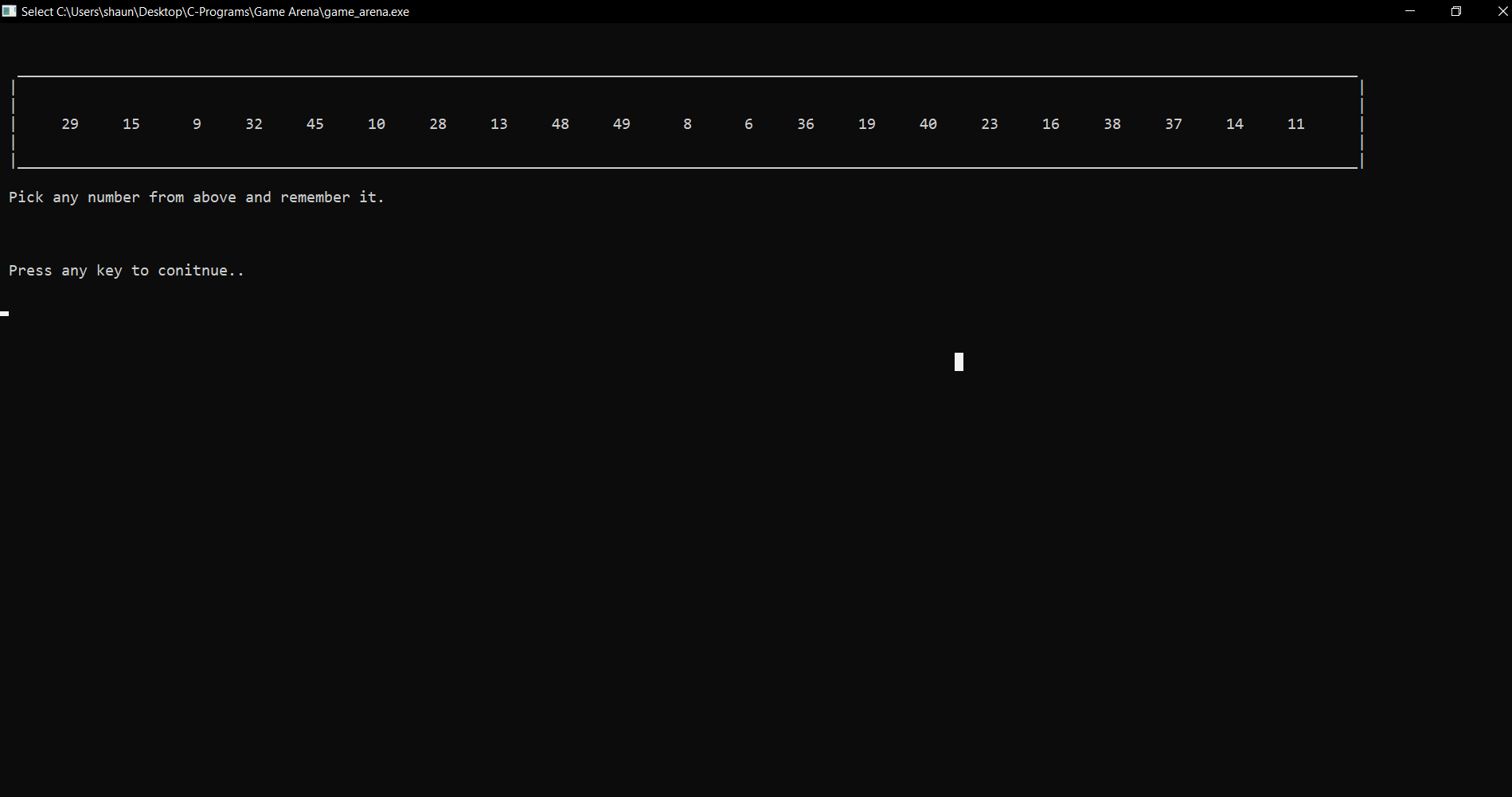


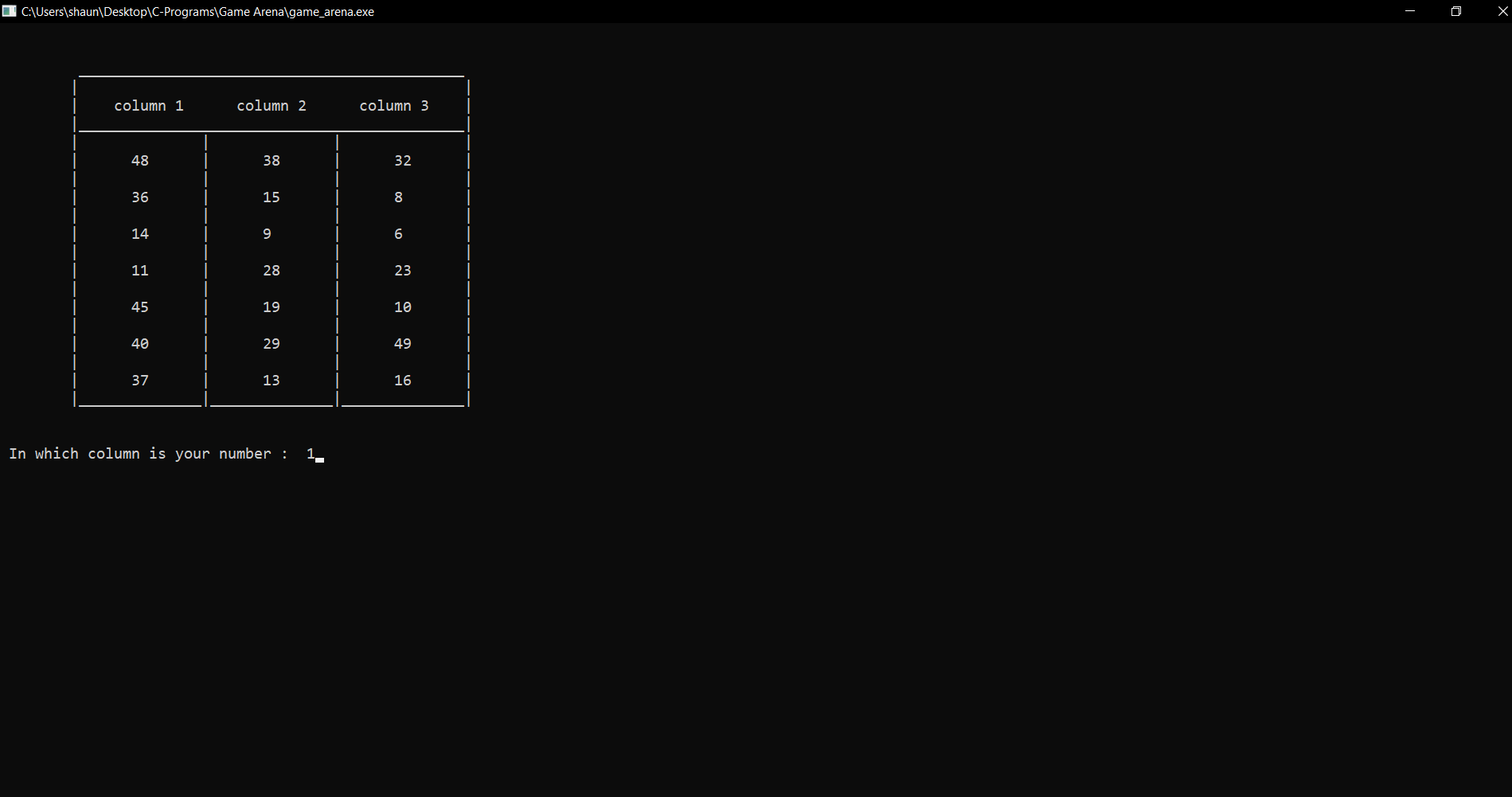


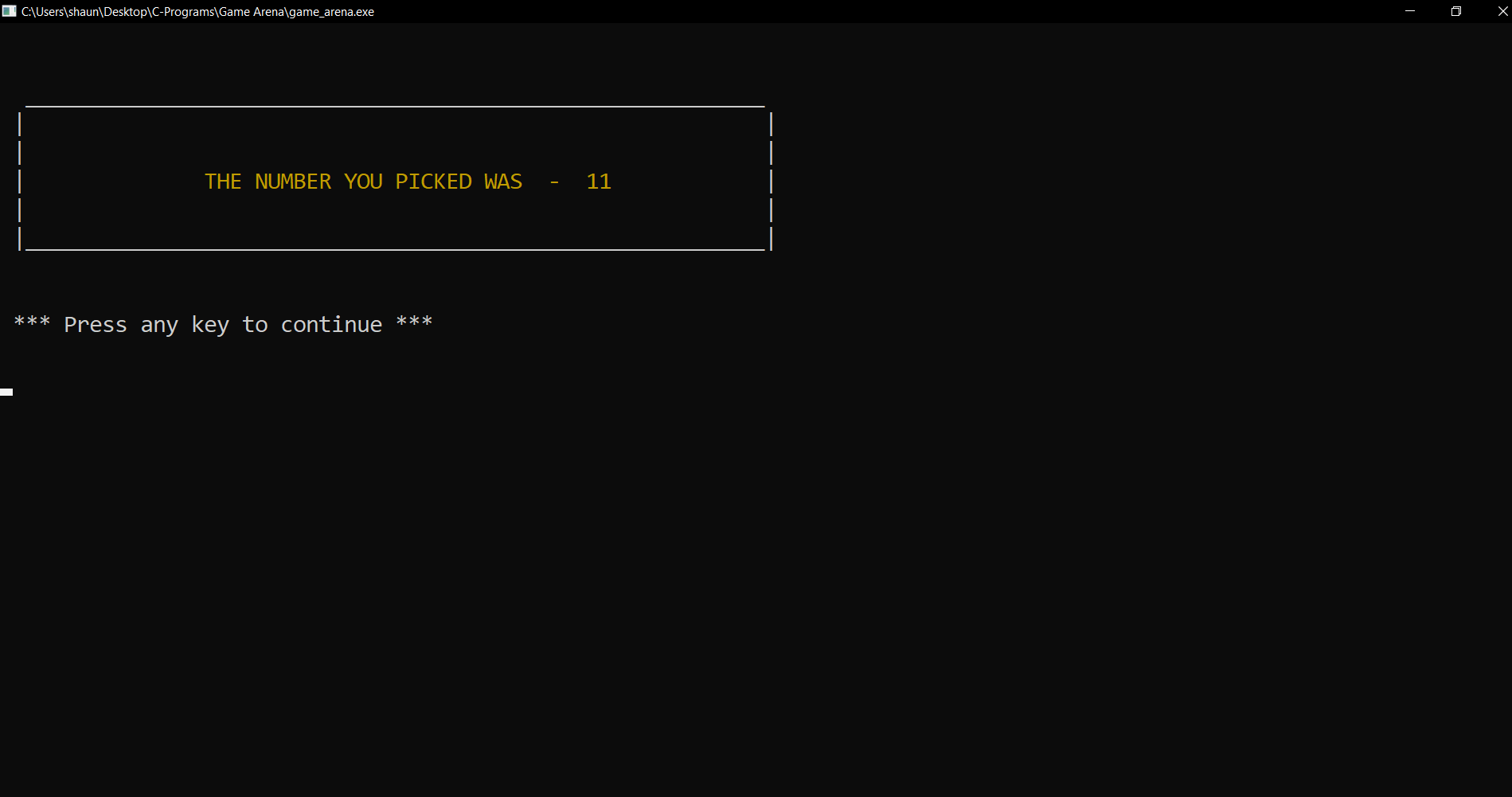


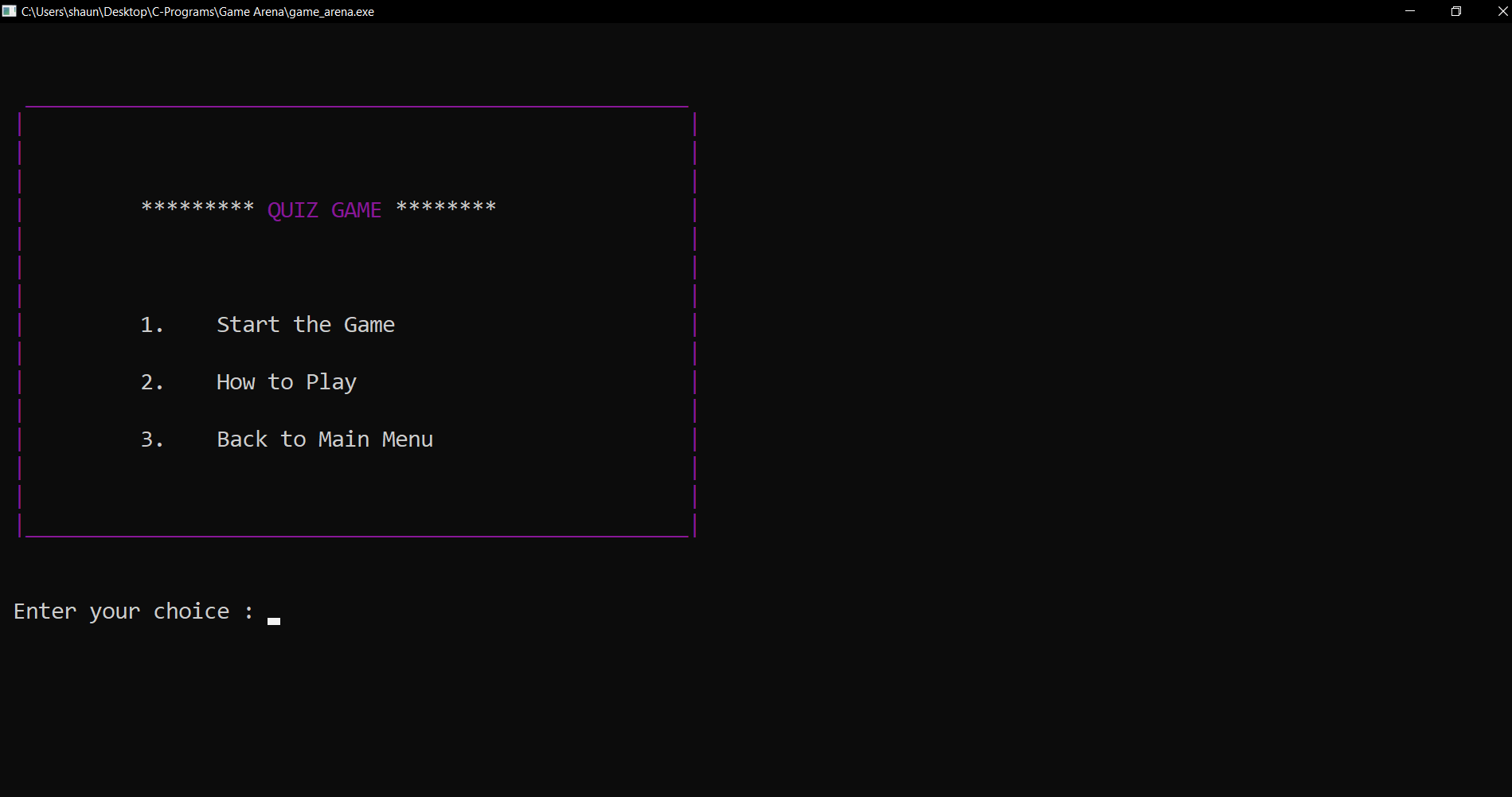


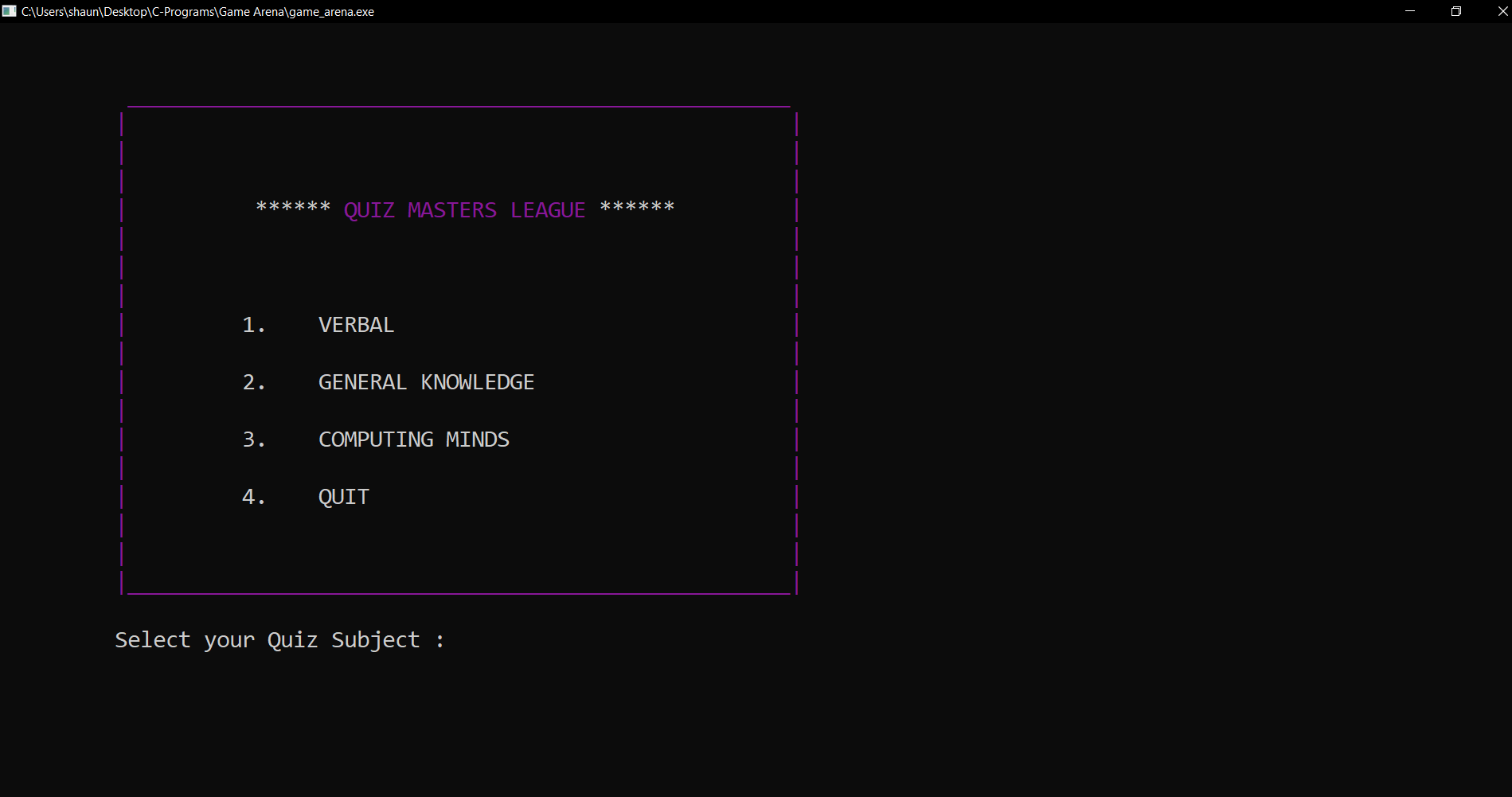


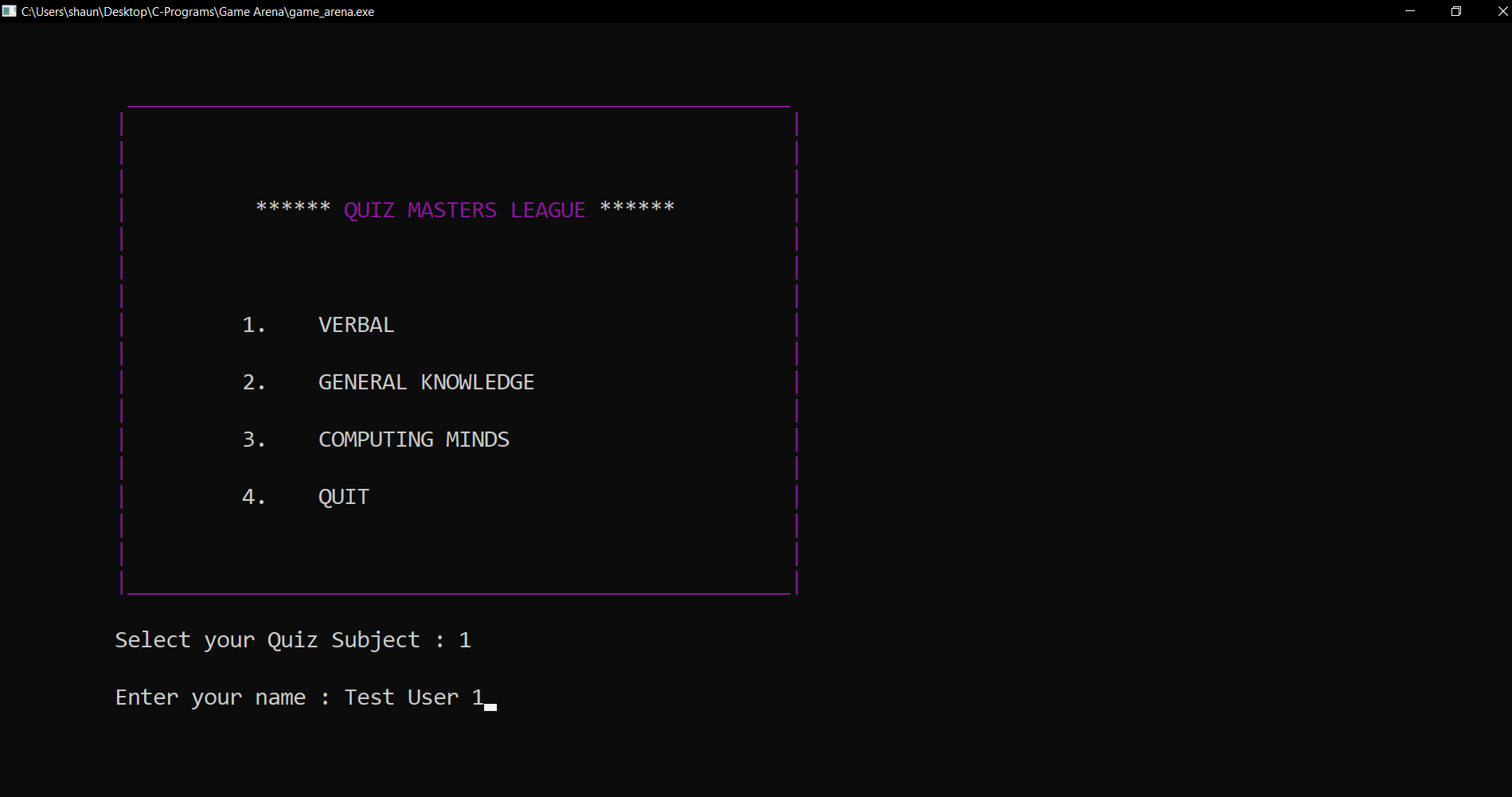


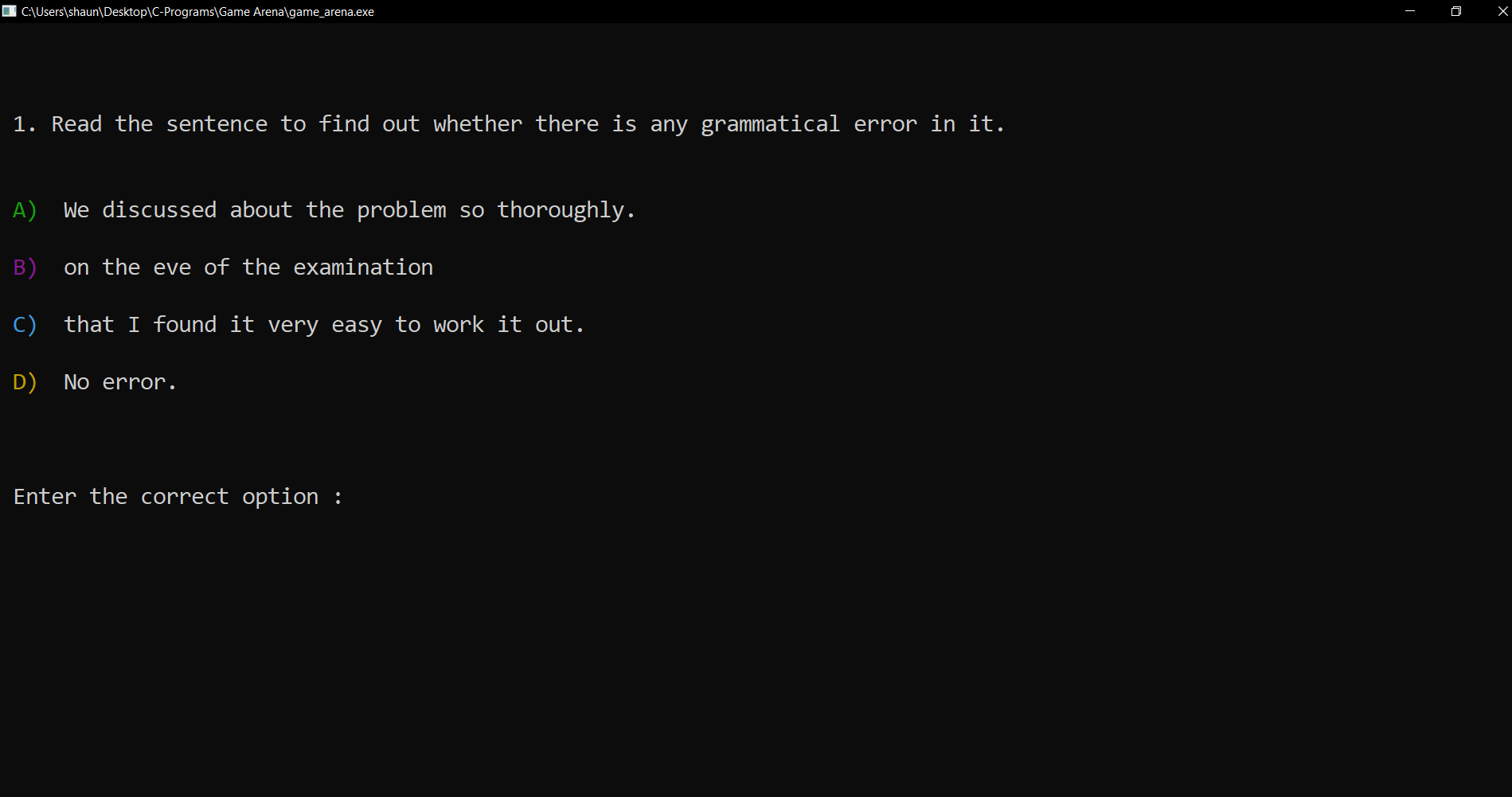


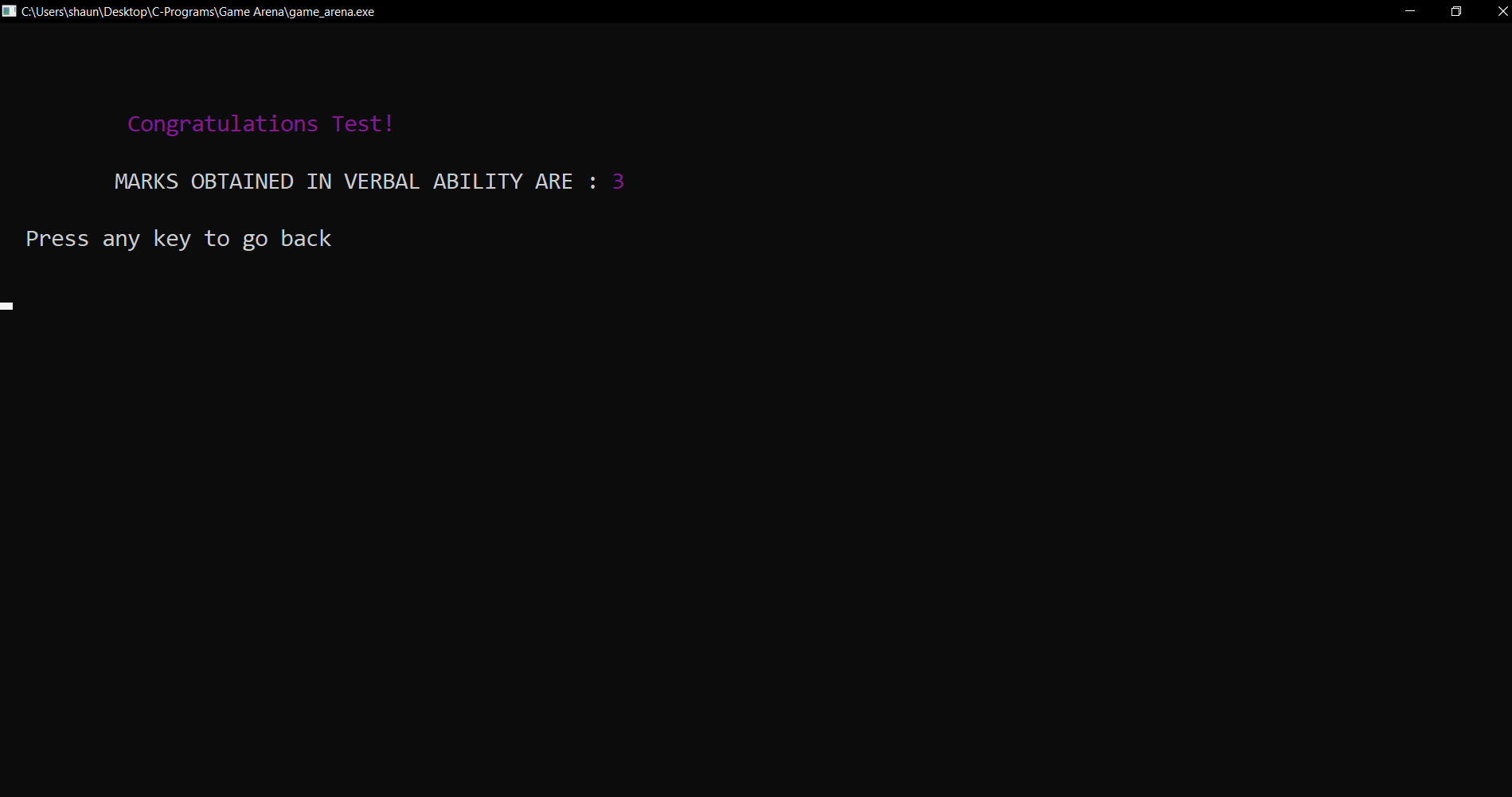


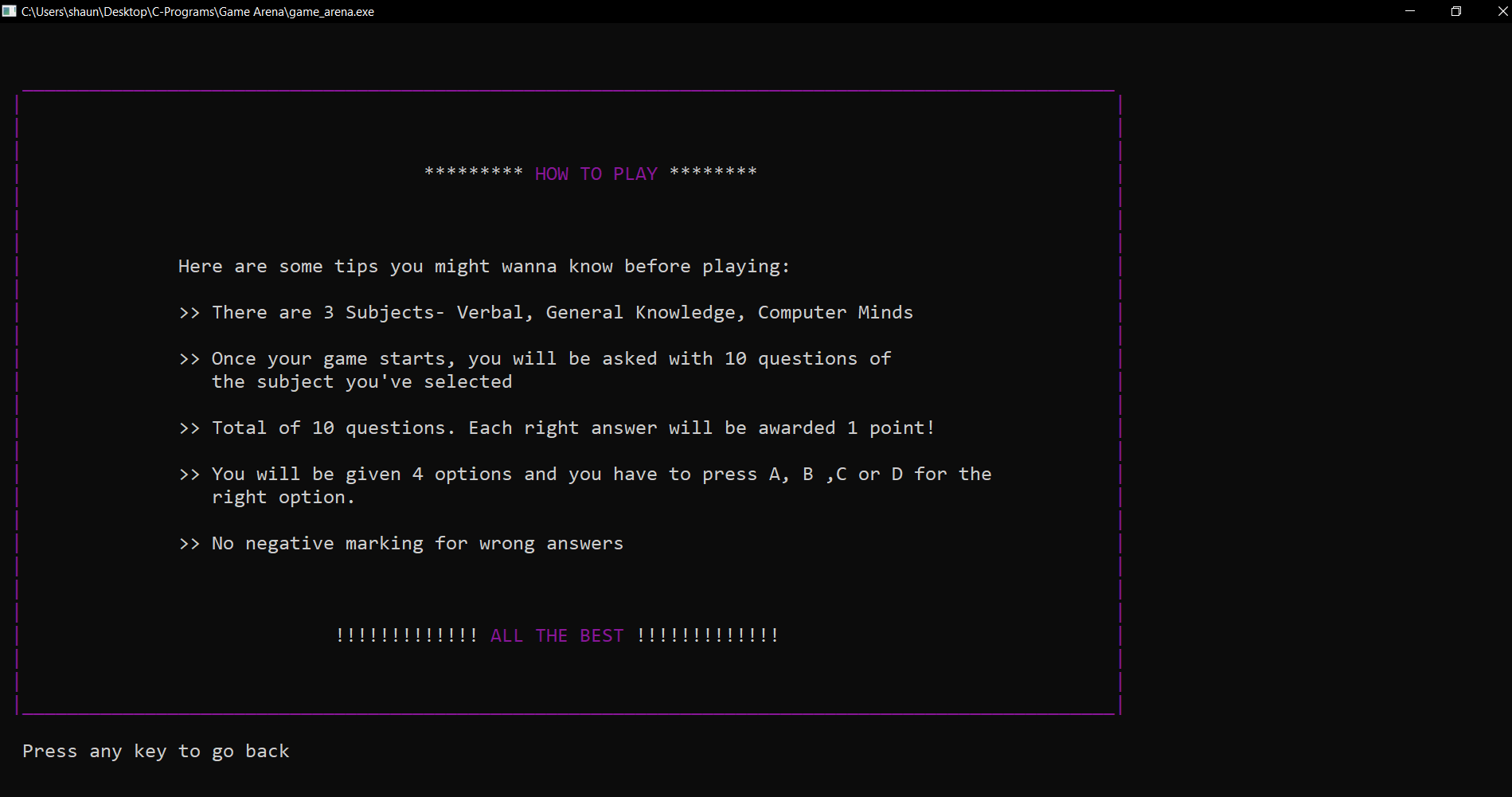


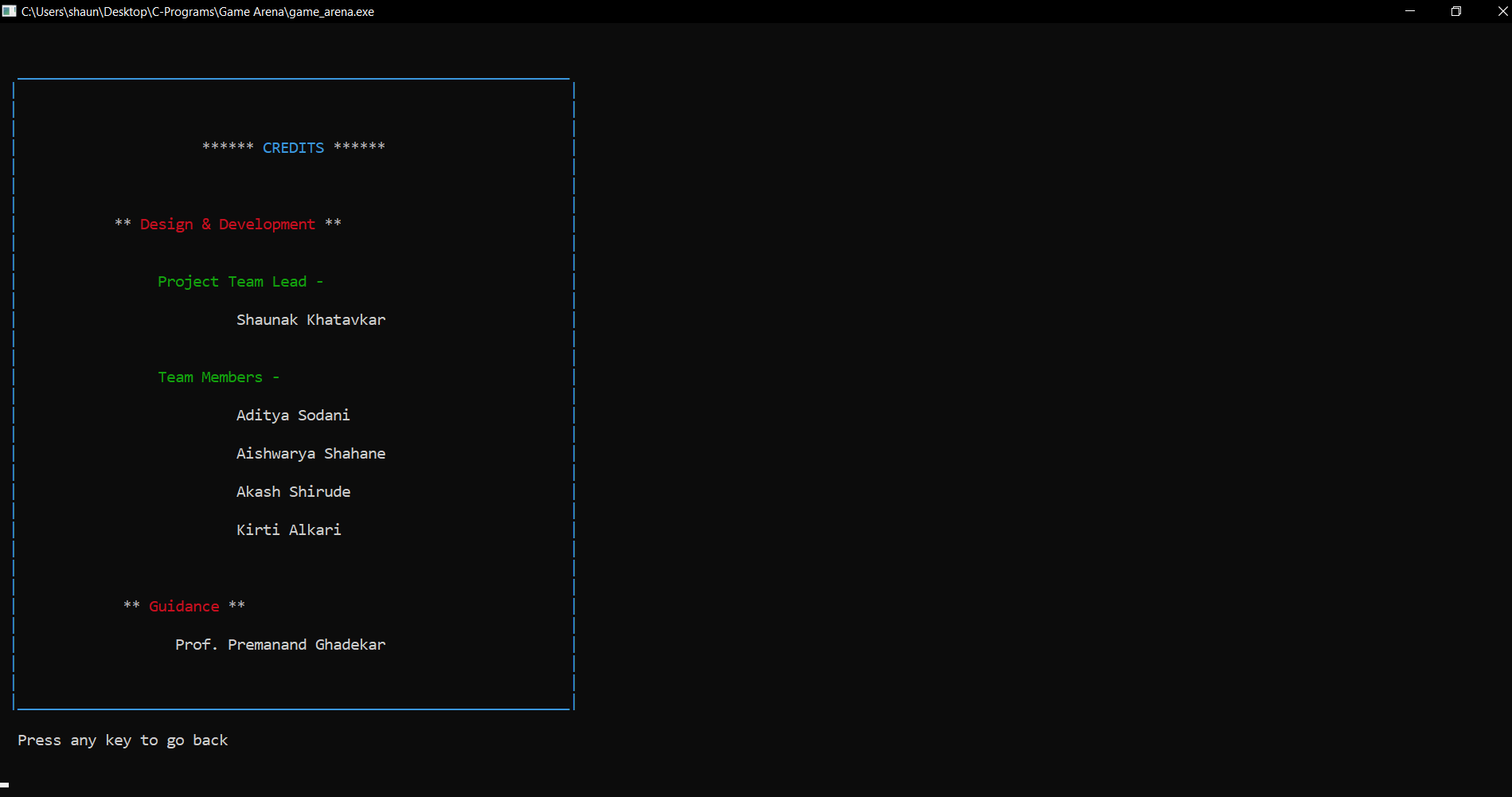












**CONCLUSION:**

For all of its advantages coding in C programming is something of a double-edged sword. Although when written in ‘C programming’ it has multiple benefits like easy implementation and less complexity also zero maintenance we cannot ignore the fact that if we use other languages for the same Gaming Arena it can be made more interactive with other OOPS features including exception handling, runtime error detection, etc.

There are many advantages of this program as it contains various features like

A] It is actually a user friendly software as its easy to use by just following the instructions which are appeared on the screen

B] This program contains educational as well as entertainment content.

C] One can easily use these games to enhance his/her knowledge in c programming , verbal ability computer skills and general knowledge

D] Playing games like Monty hall and 21 number games will be entertainment as they are producing diff result for each input.

Other programming languages have the same advantages as C, but with more features, they have a sharp learning curve that makes them less approachable by trained programmers.

As we know that , no any program can be 100 % efficient and reliable, this System is not sharply a graphical user interface.There is just use of some text color and borders

This projects aim was to build a combined system which will contain educational as well as entertainment content.and hence the aim is achieved.

**REFERENCES**

[1] <https://en.wikipedia.org/wiki/Word_search>

[2] <https://en.wikipedia.org/wiki/Monty_Hall_problem>

[3] <https://www.tutorialspoint.com/cprogramming/index.htm>

[4] <https://code.visualstudio.com/>