“QUIZANDER” A Text Based multiplayer Game

A Report submitted in partial fulfilment of the requirement for the course of

Minor Project – I

In third year – fifth semester of

B-Tech computer science Specialization in

Banking, financial services and insurance.

Under the guidance of

Mr. Alind

Assistant Professor (Senior Scale), SOCS

Department of Virtualization.

By

Name	Roll no	Sap Id	Branch
Aditya Sharma	R133218002	500069141	B-tech CSE BFSI
Rishav Singh	R133218035	500071526	B-tech CSE BFSI
Ritul Priya	R133218020	500069262	B-tech CSE BFSI

UNIVERSITY WITH A PURPOSE

SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF PETROLEUM &ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007

September -2020

Table of contents

Sr. No.	TITLE	Page
01	Abstract	3
02	Introduction	3-7
03	Motivation	7
04	Problem Statement	7
05	Literature Review	8-9
06	Objective	9
07	Methodology	9-13
08	Implementation	14-19
09	Concept Used	20
10	Challenges Faced	20
11	Plan of Work	22
12	System Requirements	23
13	Work done so far	23
14	References	24

Minor I

PROJECT TITLE: “QUIZANDER” A Text Based Multiplayer Game

ABSTRACT

We are making a text based multiplayer game. This game is a sequence of knowledge and fun. This game requires no prior experience. It can be played by anyone. This game can be played in both single player mode and multiplayer mode. There will be five exciting categories. The player can choose whichever kind it wants to take part in. Each category has three rounds that are, the initial round, the intermediate round and the final round.

The user can only move to succeeding round until and unless it passes the passing criteria that are different in every round. There will be a time limit. For each correct answer, a point will be credited. There will be a negative marking only in multiplayer mode.

In the single-player mode for each correct answer within a time frame, the player would be rewarded with a point. In the multiplayer mode, the marking system would be based on the first correct answer i.e., anyone who answered correctly in less time would be given more marks as compared to the other player and if the player will give a wrong answer then there will be a negative marking.

The user will get two hints in each round. The highest scorer will be entitled by “QUIZHILIC”.

This will be an interesting journey where the fun in each round will be unique and exciting.

INTRODUCTION

Earlier, games were less creative and ordinary. But, as the era changes evolution of these games is praiseworthy. Playing trivia games helps in developing fluid intelligence. We are making a text-based game which will be an exciting past-time.

Firstly, by removing the boring content so, in this game, we are adding five different interesting categories which will not make the user feel weary.

Each division will have three levels which will make the user’s mind blend.

There will be a time limit for every question, so we are adding a timer by using a sleep() function which gives a simple way to make the program wait for short interval from windows.h header file.

Earlier all the questions were displayed on the same page but in our game, one question will be displayed in one window screen. This will happen by using the system(“cls”) function. This function helps in clearing the screen.

We are using the stdlib.h library file which will help us in generating the random number sequence for the questions, string manipulations.

We are also using stdio.h header file which contains input/output functions, C declaration and many more.

This game can be played in both the modes that are, the single-player mode and the multiplayer mode.

We will be also using the file handling for storage of high scores. So that the scores won’t get lost.

Evolution in Gaming Industry

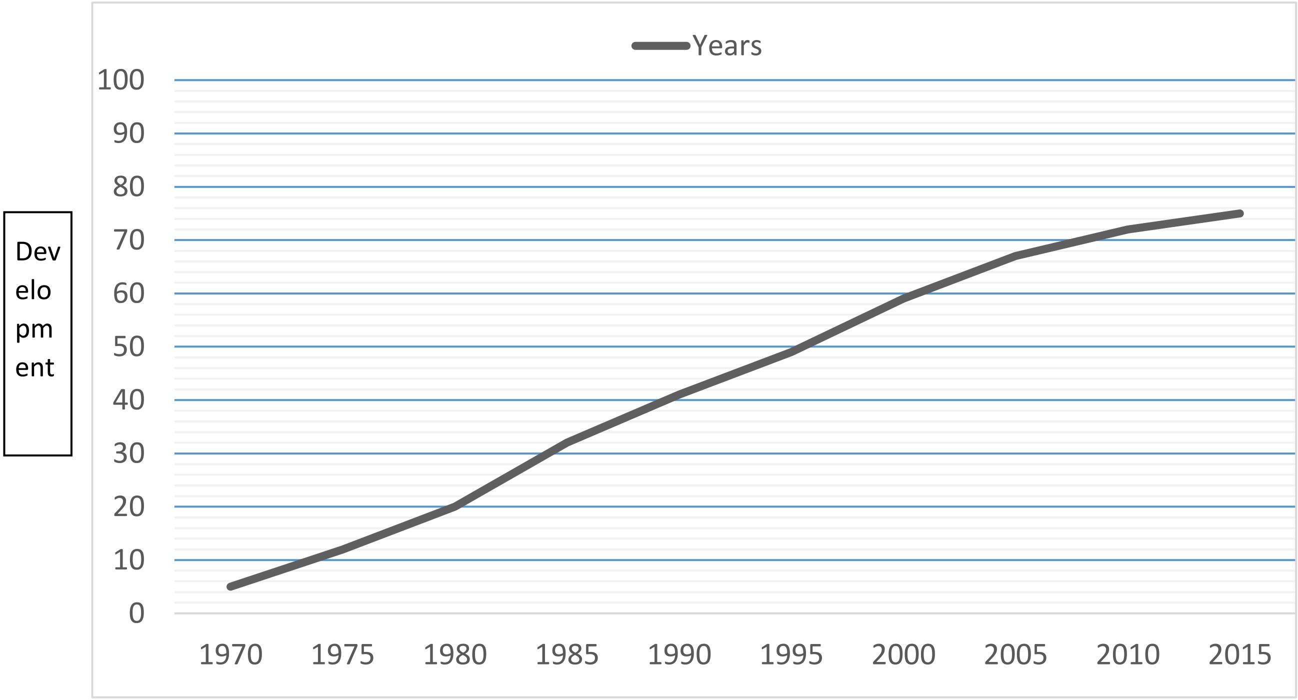


Fig.1:-Evolution in gaming industry

The above graph is between development in percentage and years. Basically, this graph shows that how gaming industry is increasing year by year.

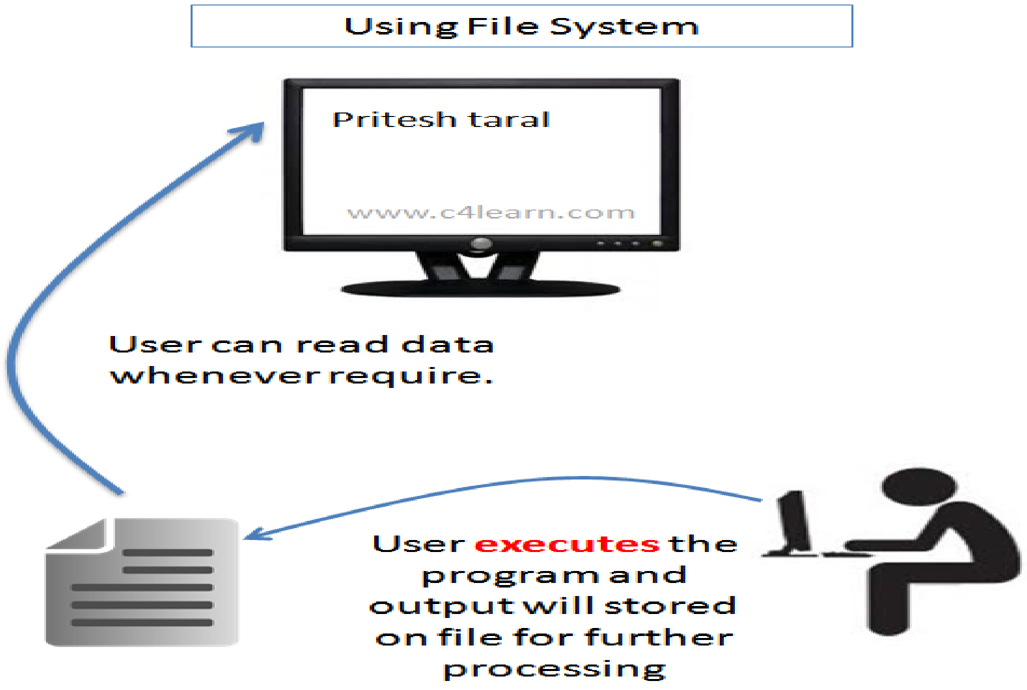

Using File System
Pritesh taral
www.c4learn.com
User can read data whenever require.
User executes the program and output will stored on file for further processing

Fig. 2:-File Handling is used for Storing the high scores in the text based game.

The Above Figure is showing how a user can read, delete, update and create the files stored on the local file system through the execution of program.

The 5 categories are:

1:- 10 second observer Challenge

2:- Polling Round Fire

3:- Match the following

4:- Getting the ways out

5:- Ultimate Fusion

- In 10 Second Challenge part, the player will be displayed 30 objects name on the screen for 10 seconds and after that screen will be cleared and the player has to write the object name which they have observed.
- In Polling Round Fire, the player will be given a question which they have to determine how much percentage the public of India support the answer.
- In Match the Following part, the player will be provided with two columns in which they have to write the correct sequence.
- In Getting the ways out part, the player have to count the number of ways he/she can reach the destination from the source that is, moving from the top leftmost cell to the bottom rightmost cell. The user can only move in right and downward direction.

- The Ultimate Fusion part is the mixture of different categories, including the categories mentioned above and also different categories like jumble words, riddles etc.

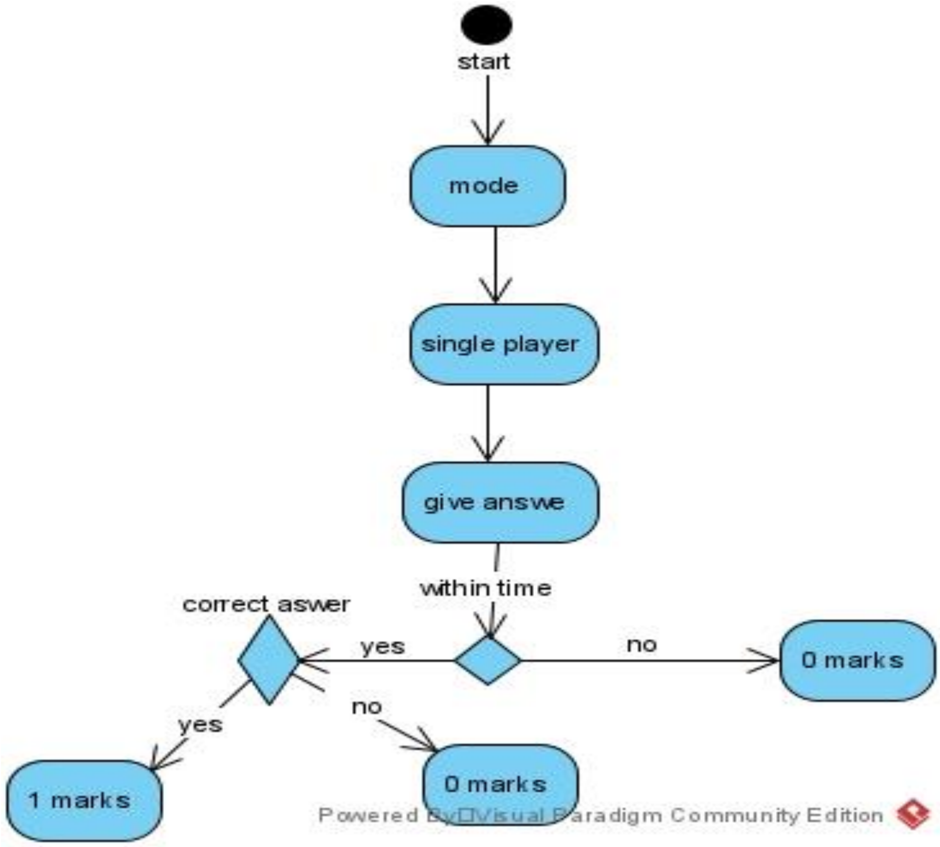
Marking Rule:-

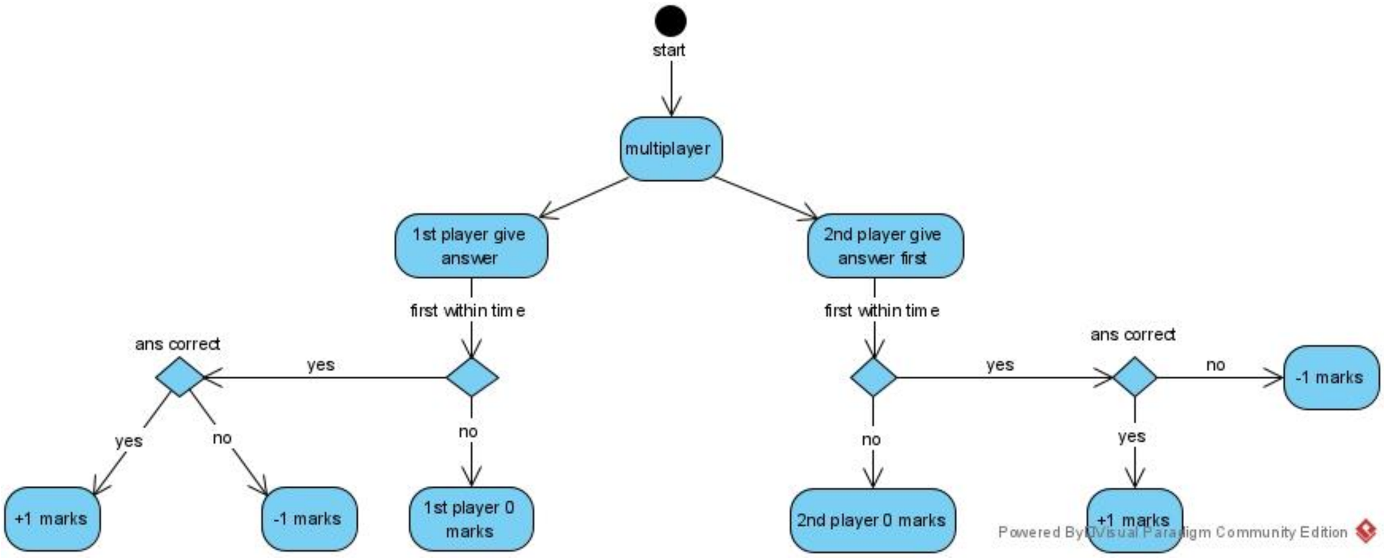
If the player answers correctly within the given time interval they would be rewarded by +1 point and there will be no negative marking in the single-player mode. In Multiplayer mode both the players will get the question at the same time where the player who attempts first within the time interval and gives the correct answer will be rewarded by +1 point and if the answer will be wrong, that player will get -1 point. If both the players give correct answer within the given time interval, the player who has attempted first will be rewarded by +1 point and the second player will be rewarded by +0.5 point. There will be three rounds.

A:-**Basic Round** (Total 7 Question) 30 seconds for each question. You have to score at least 5 Points in the basic round to enter in the Intermediate Round.

B:-**Intermediate Round** (Total 10 Question) 40 seconds for each question. You have to score at least 8 points in the intermediate round to enter in the Final Round.

C:-**Final Round** (Total 12 questions) 50 seconds for each question. You have to score at least 10 points to in order to check if you gain the entitlement of "**QUIZHILIC**"





MOTIVATION

- We are building an interesting game. This is a trivia quiz play which involves many sectors of learning. We are using C language for our project. As we know C language is the mother of all languages. By creating this project, it will help us in learning the basic understanding of C programming. We will be learning about the software engineering part like, software models, their stages and many more. This is basically for the training and learning purpose which will enhance our knowledge and skills for future projects.
- Building and implementing different algorithm for this project.
- Studying and implementing is the best way to learn and understand any concept.

PROBLEM STATEMENT

As most text-based games are in single player mode and players miss the exciting catches. So we are building a text based multiplayer game in a limited time frame, where the questions will be like a ride and will make the user’s mind blend.

LITERATURE REVIEW

- In paper[1] named **Counting to Explore and Generalize in Text-based Games** by **Xingdi Yuan * Marc-Alexandre Cotˆ e´ * Alessandro Sordoni Romain Laroche Remi Tachet des Combes Matthew Hausknecht Adam Trischler** stated that
  Text-based games like Zork (Infocom, 1980) are complex, interactive simulations. They use natural language to describe the state of the world, to accept actions from the player, and to report subsequent changes in the environment. The player works toward goals which are seldom specified explicitly and must be discovered through exploration. The observation and action spaces in text games are both combinatorial and compositional, and players must contend with partial observability, since descriptive text does not communicate complete, unambiguous information about the underlying game state.
- In paper[2] named **Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning** by **Prithviraj Ammanabrolu** School of Interactive Computing Georgia Institute of Technology Atlanta, GA ,**Mark O. Riedl** School of Interactive Computing Georgia Institute of Technology Atlanta, GA states that
  Natural language communication can be used to affect change in the real world. Text adventure games, in which players must make sense of the world through text descriptions and declare actions through natural language, can provide a stepping stone toward more real-world environments where agents must communicate to understand the state of the world and indirectly affect change in the world. Text adventure games are also useful for developing and testing reinforcement learning algorithms that must deal with the partial observability of the world.
- In paper[3] named **FOOTPAD-BASED GAME AND GAMING SYSTEM** by **Molly Casey**, **Denver**, CO (US); **Patricia L. Christen, Piedmont**, CA (US); and the team states that
  A keypad mat may be used to play a variety of games, particularly games involving text messaging, and may be used to send text messages as part of the game. For example, a text messaging game system may include a key pad mat for typing text messages by foot movement that is configured as a phone keypad. The phone keypad mat may be configured as a foot or “dance' mat that looks like a cell phone keypad. The game controller may be a small game box that is attached to (or integral with) the mat. The game controller and/or the mat can then connect to an output device. Such a television, computer, cell phone, or other display.

- In paper [4] named **Implementation of Pattern Matching Algorithm for Portable Document Format** by Anton Yudhana Department of Electrical Engineering Ahmad Dahlan University Yogyakarta, Indonesia, Sunardi Department of Electrical Engineering Ahmad Dahlan University Yogyakarta, Indonesia, Abdul Djalil Djayali Master of Informatics Engineering Ahmad Dahlan University Yogyakarta, Indonesia states that

Rabin-Karp algorithm is a string-matching algorithm that uses hash functions as a comparison between the search string (m) and substring in a text (n). The Rabin-Karp algorithm is based on the fact that if two strings are equal then the hash value must be the same. But there are two problems that arise from this, the first problem is that there are so many different strings, this problem can be solved by assigning multiple strings with the same hash value. The second problem is not necessarily a string that has the same hash value matching to overcome it for each string that is assigned to do string matching by BruteForce [1], [3]

OBJECTIVE

1. Designing the Main Window.
2. Creating an interface for the users to choose from different categories.
3. Inclusion of time for each question.
4. Displaying the questions randomly using random generator.
5. Designing the interface for multiplayer mode on a single screen.
6. Integration of different categories.
7. Implementing the algorithms for each round.
8. Storing their scores in order to check the high scores.

METHODOLOGY

We are using iterative waterfall model for our project. As, this model is easily understandable and the main profit of this model is that it allows us to go back on the previous phase and change the requirements and do some modification if necessary.

As iterative model has six stages:

- Feasibility study: We have minimised the total lines of code in order to light the workability of the game and to reduce the run time and compilation time.
- Requirement analysis and specification: We have collected the information and database required in different categories.
- Design: We have designed the main window screen and single player mode interface.
- Coding and unit testing: We have done the coding part of single player mode and performed unit testing for different categories.
- Integration and system testing: The integration and system testing will be done once we complete the coding of both the mode.

- Maintenance: This will be easily maintainable as we can go to the previous phase and do the changes if required.

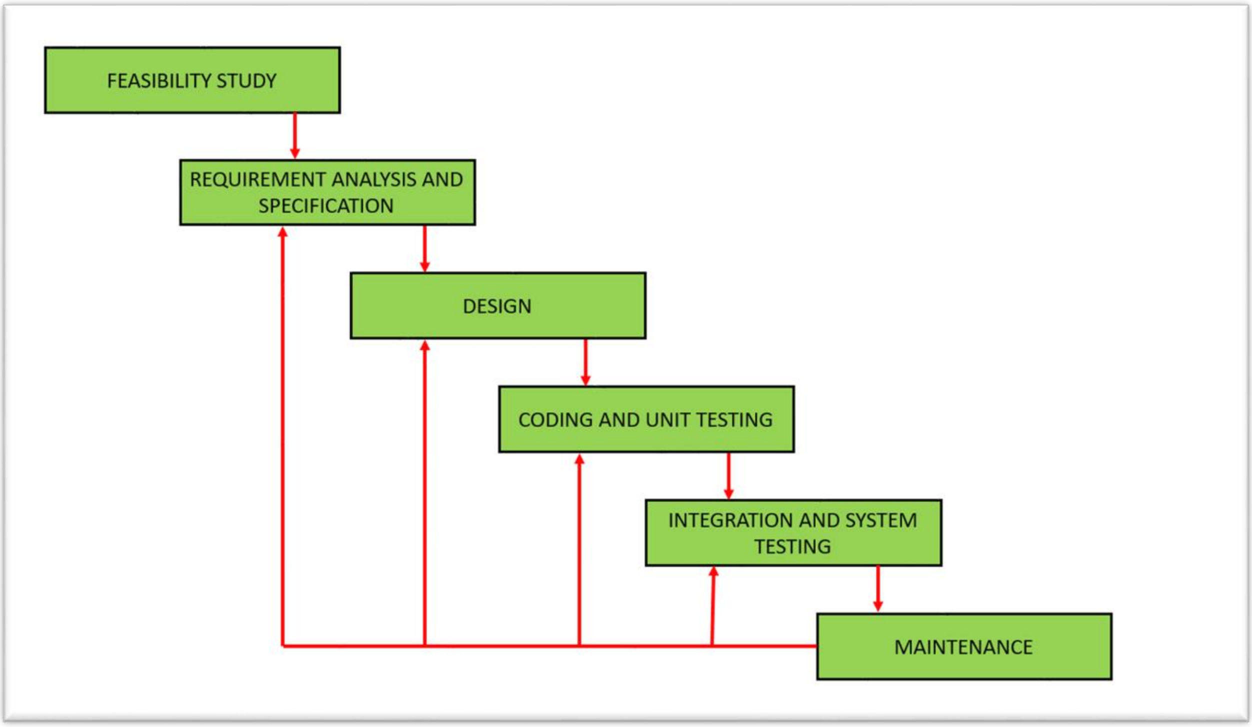

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Fig 1. Feedback paths introduced by the iterative waterfall model as shown in the figure.

ALGORITHM

Count number of ways to reach destination in a Maze Algorithm:-

Given a maze with obstacles, count number of paths to reach rightmost-bottommost cell from topmost-leftmost cell. A cell in given maze has value -1 if it is a blockage or dead end, else 0.

The idea is to modify the given grid[][] so that grid[i][j] contains count of paths to reach (i, j) from (0, 0) if (i, j) is not a blockage, else grid[i][j] remains -1.

We can recursively compute grid[i][j] using below

Formula and finally return grid[R-1][C-1]

```
// If current cell is a blockage
  if (maze[i][j] == -1)
      maze[i][j] = -1; //  Do not change
 // If we can reach maze[i][j] from maze[i-1][j]
```

```
  // then increment count.
  else if (maze[i-1][j] > 0)
      maze[i][j] = (maze[i][j] + maze[i-1][j]);
 // If we can reach maze[i][j] from maze[i][j-1]
  // then increment count.
  else if (maze[i][j-1] > 0)
maze[i][j] = (maze[i][j] + maze[i][j-1]);
```

Rabin-Karp Algorithm for Pattern Searching:-

Rabin-Karp algorithm also slides the pattern one by one. But unlike the Naive algorithm, Rabin Karp algorithm matches the hash value of the pattern with the hash value of current substring of text, and if the hash values match then only it starts matching individual characters. So Rabin Karp algorithm needs to calculate hash values for following strings.

1) Pattern itself.
2) All the substrings of text of length m.

Since we need to efficiently calculate hash values for all the substrings of size m of text, we must have a hash function which has following property.
Hash at the next shift must be efficiently computable from the current hash value and next character in text or we can say hash(txt[s+1 .. s+m]) must be efficiently computable from hash(txt[s .. s+m-1]) and txt[s+m] i.e., hash(txt[s+1 .. s+m])= rehash(txt[s+m], hash(txt[s .. s+m-1])) and rehash must be O(1) operation.The hash function suggested by Rabin and Karp calculates an integer value.

RABIN-KARP-MATCHER (T, P, d, q)

```
n ← length [T]
m ← length [P]
h ← $d^{m-1}$ mod q
p ← 0
$t_0$ ← 0
for i ← 1 to m
```

```
do p ← (dp + P[i]) mod q
$t_0$ ← ($dt_0$+T [i]) mod q
for s ← 0 to n-m
do if p = $t_s$
then if P [1.....m] = T [s+1.....s + m]
then "Pattern occurs with shift" s
If s < n-m
then $t_{s+1}$ ← (d ($t_s$-T [s+1]h)+T [s+m+1])mod q
```

Marking Under Time Algorithm:-

Marking under time algorithm is a marking system based on the time interval scheduling algorithm. This algorithm states that if a player attempted the following question within the time frame. Their timing will be compared with respective players who have attempted till the time. If the player has attempted in the least time then there answer will be matched with the correct option and if it is correct they will be rewarded by +1 point and if it is incorrect the player will get -1 point. If the player time is more than any other attempted person, then again a condition will be checked whether the answer of the player who attempted first is correct or not. If the answer given by the player who attempted first is correct then the respective player will be rewarded by +0.5 point where as the player who attempted first answer is wrong then the respective player will be rewarded as +1 point .Player who will not attempt the question will be given no marks.

T1 – Total time of Player 1.

T2 – Total time of Player 2.

P1 – Player 1.

P2 – Player 2.

C1 – Choice of 1^{St} Player.

C2 – Choice of 2^{nd} Player.

START ALGORITHM

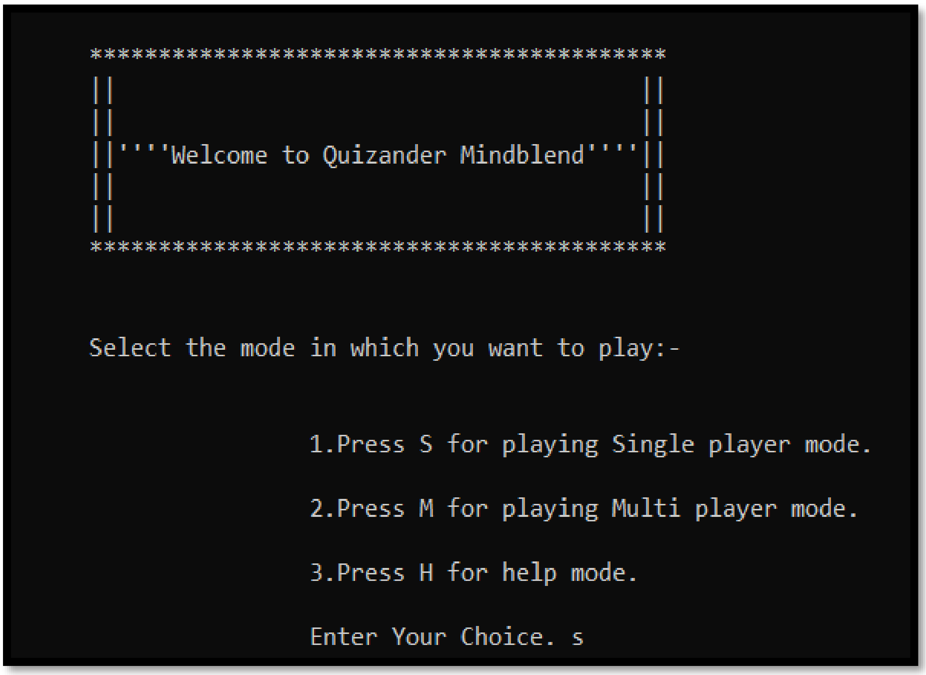
```
int MARKING(T1, T2, P1, P2, C1, C2)
P1 =C1, P2 = C2
IF(T1<T2)
  THEN
```

```
    IF (C1 = correct && C2 = correct)
THEN P1 marked as = +1 point
P2 marked as = +0.5 point
    ELSE IF (C1 != correct && C2 = correct)
        THEN P1 marked as = -1 point
                P2 marked as = +1 point
    ELSE IF (C1 = correct &&C2 != not attempted)
        THEN P1 marked as = +1 point
                 P2 marked as = 0 point
   ELSE IF (C1 != correct && C2 != correct)
        THEN P1 marked as = -1 point
                 P2 marked as = -1 point
   ELSE (C1 = not attempted && C2 = not attempted)
        THEN P1 marked as = 0 point
                P2 marked as = 0 point
OTHERWISE (T2<T1)
  THEN
     If (C1 = correct && C2 = correct)
         THEN P1 marked as = +0.5 point
                 P2 marked as = +1 point
     ELSE IF (C1 = correct &&C2 != correct)
        THEN P1 marked as = +1 point
                P2 marked as = -1 point
    ELSE IF (C1 = not attempted && C2 = correct)
       THEN P1 marked as = 0 point
           P2 marked as = +1 point
   ELSE IF (C1 != correct && C2 != correct)
        THEN P1 marked as = -1 point
           P2 marked as = -1 point
   ELSE (C1 = not attempted && C2 = not attempted)
        THEN P1 marked as = 0 point
                P2 marked as = 0 point
```

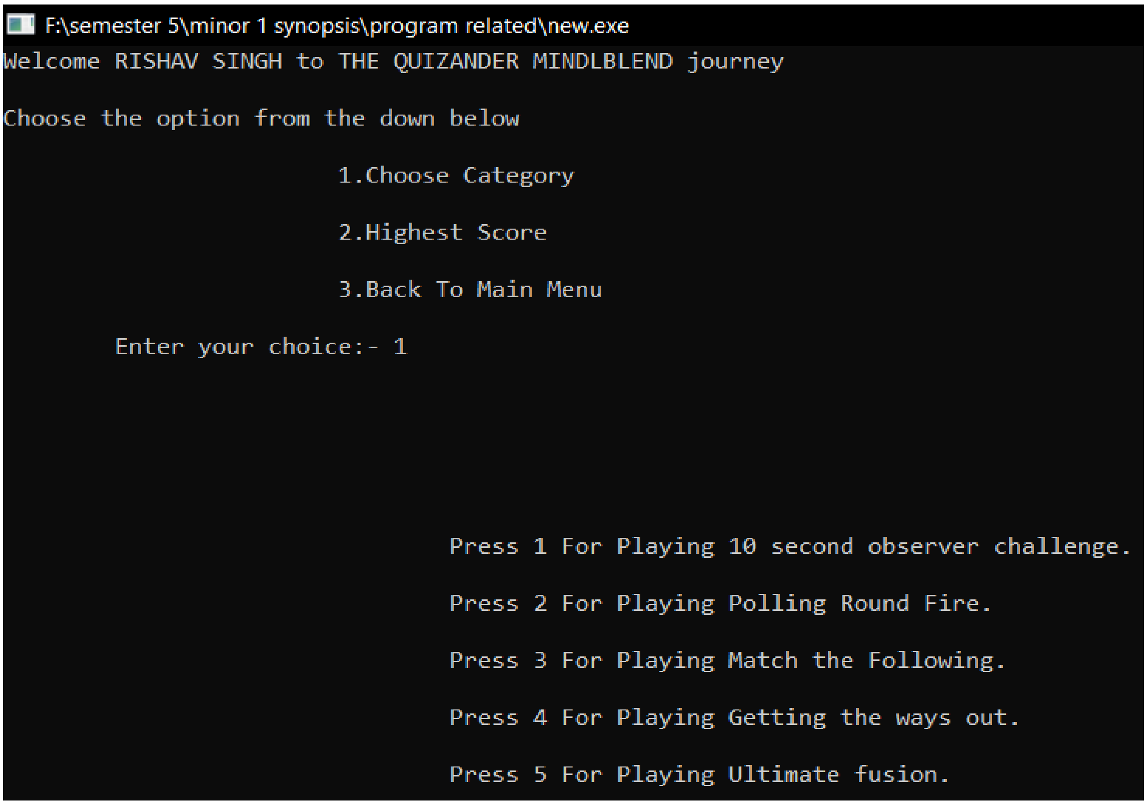
Pseudocode For Main Window Screen

```
Step 1: Import all the Header Files.
Step 2: Declare Main function.
            printf("Welcome Screen")
Step 3: Display Choice for player.
             S = Single Player
            M= Multiplayer
            H = Help
Step 4: Declare function for Single Player.
            Declare function for Multiplayer.
            Declare function for Help Mode.
            printf("Write Name")
            Declare Choices for Player Mode
           {
                1. Choose Category
                2. Highest Score
                3. Back to Main menu
           }}
Stop}
```

Output

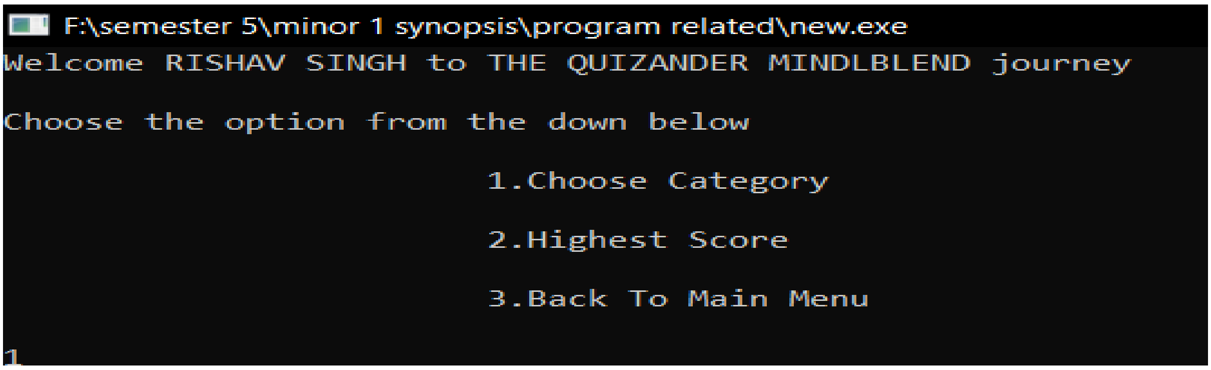


```
******************************************
||                                      ||
||                                      ||
||''''Welcome to Quizander Mindblend''''||
||                                      ||
||                                      ||
******************************************


Select the mode in which you want to play:-


                1.Press S for playing Single player mode.

                2.Press M for playing Multi player mode.

                3.Press H for help mode.

                Enter Your Choice. s
```



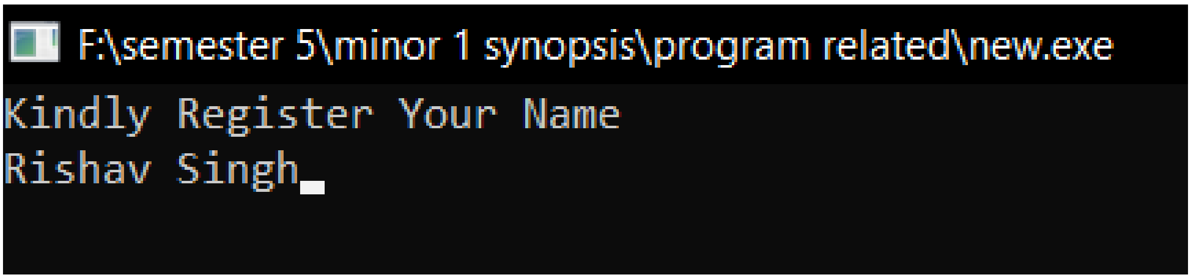
F:\semester 5\minor 1 synopsis\program related\new.exe

```
Welcome RISHAV SINGH to THE QUIZANDER MINDLBLEND journey

Choose the option from the down below

                        1.Choose Category

                        2.Highest Score

                        3.Back To Main Menu

        Enter your choice:- 1




                                Press 1 For Playing 10 second observer challenge.

                                Press 2 For Playing Polling Round Fire.

                                Press 3 For Playing Match the Following.

                                Press 4 For Playing Getting the ways out.

                                Press 5 For Playing Ultimate fusion.
```



F:\semester 5\minor 1 synopsis\program related\new.exe

```
Welcome RISHAV SINGH to THE QUIZANDER MINDLBLEND journey

Choose the option from the down below

                        1.Choose Category

                        2.Highest Score

                        3.Back To Main Menu

1
```



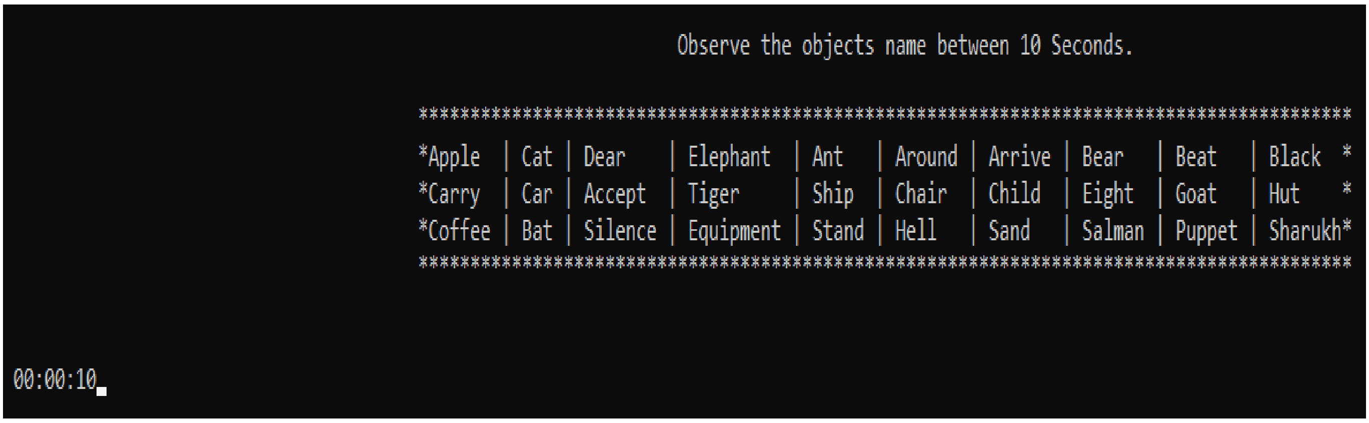
F:\semester 5\minor 1 synopsis\program related\new.exe

```
Kindly Register Your Name
Rishav Singh_
```

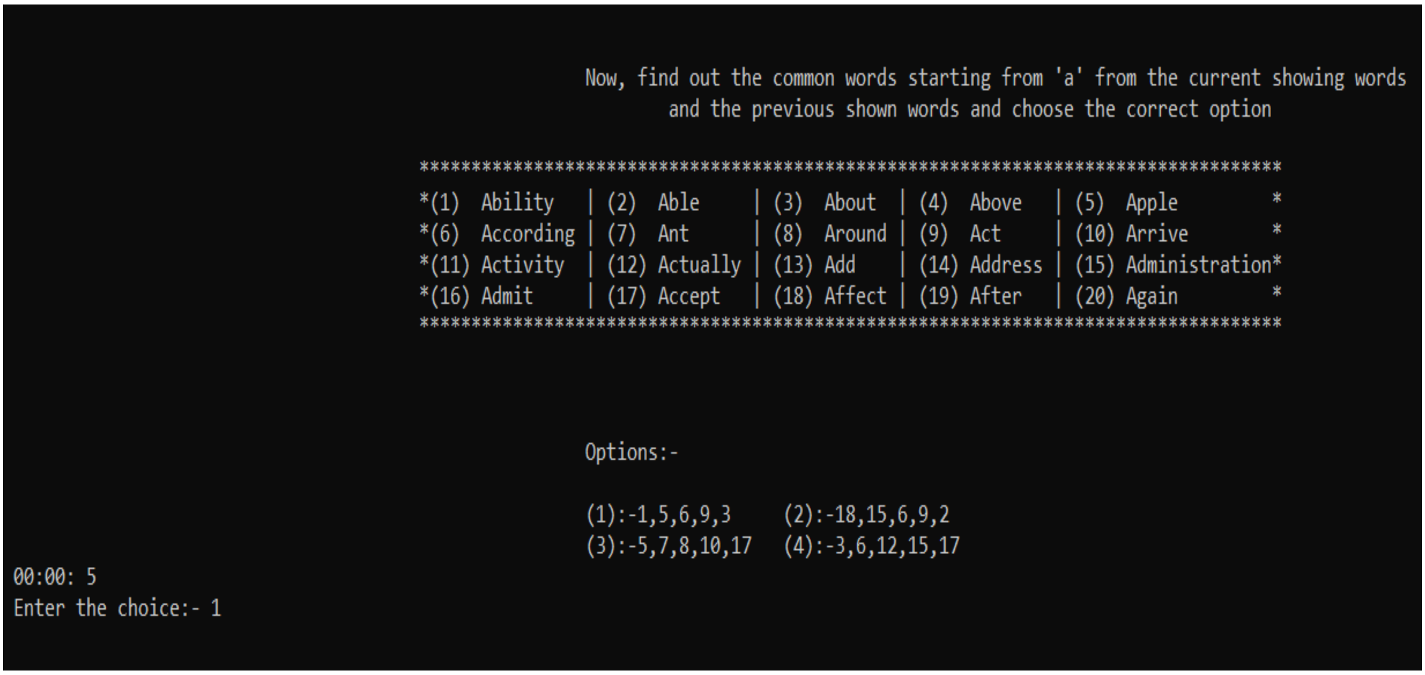
Pseudocode For 10 Second Observer

```
Step 1: Import all header files.
Step 2: Declare the object function by:
            int object(*name);
Step 3: Declare main function
Step 4: Call the object function (object())
            in the main:
            print the welcome screen.

Step 5: Declare the random generator function.
int object(char* name)
{
srand(time(NULL));
for(i=0;i<=n-1;i++)//n is no of question.
{
temp=rand()%n;
for(j=0;j<=i-1;j++){
if(temp==a[j])
break;}
if(j==i)
a[i]=temp;
else
i--;}
Step 6: Declare variable for the object and display
30 objects name for 10 seconds.
Step 7: while(!kbhit() && TRUE)
      {
      printf("     \r 00:00:%2d",s);
                        ms++;
                        Sleep(10);
                        if(ms==11)
}
      system("cls");
Step 8: Display questions
Step 9: Declare time function
Step 10: while(!kbhit() && TRUE)
      {
      printf("     \r 00:00:%2d",s);
                        ms++;
                        Sleep(10);
                        if(ms==31)
}                             system("cls");

Step 11 : Accepts the user input and check:
            if(answer==right)
            marks=marks+1
            else
            marks=marks+0;
Step 12: After the completion of each round:
            if(total_marks>=required_score)
                  enter to next round;
            else
                  back to main menu;
```

Output



```
                                          Observe the objects name between 10 Seconds.

                          ***********************************************************************************
                          *Apple  | Cat | Dear    | Elephant  | Ant   | Around | Arrive | Bear   | Beat   | Black  *
                          *Carry  | Car | Accept  | Tiger     | Ship  | Chair  | Child  | Eight  | Goat   | Hut    *
                          *Coffee | Bat | Silence | Equipment | Stand | Hell   | Sand   | Salman | Puppet | Sharukh*
                          ***********************************************************************************



00:00:10_
```

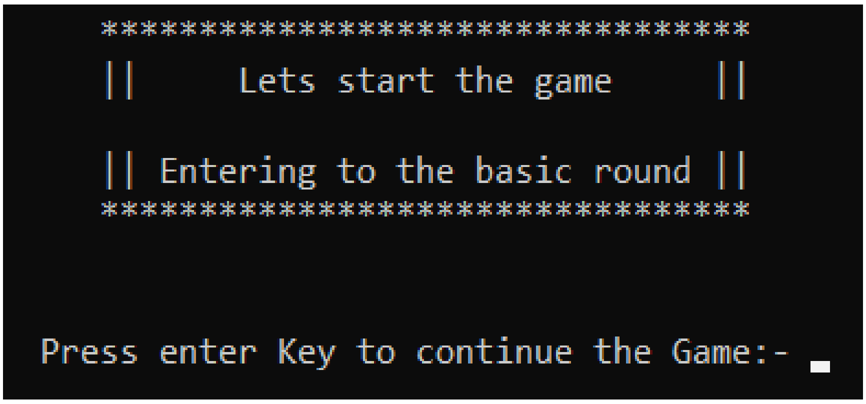


```
                                   Now, find out the common words starting from 'a' from the current showing words
                                        and the previous shown words and choose the correct option

                          ********************************************************************************
                          *(1)  Ability  | (2)  Able     | (3)  About  | (4)  Above   | (5)  Apple          *
                          *(6)  According | (7)  Ant     | (8)  Around | (9)  Act     | (10) Arrive         *
                          *(11) Activity | (12) Actually | (13) Add    | (14) Address | (15) Administration*
                          *(16) Admit    | (17) Accept   | (18) Affect | (19) After   | (20) Again          *
                          ********************************************************************************




                                   Options:-

                                   (1):-1,5,6,9,3      (2):-18,15,6,9,2
                                   (3):-5,7,8,10,17    (4):-3,6,12,15,17
00:00: 5
Enter the choice:- 1
```

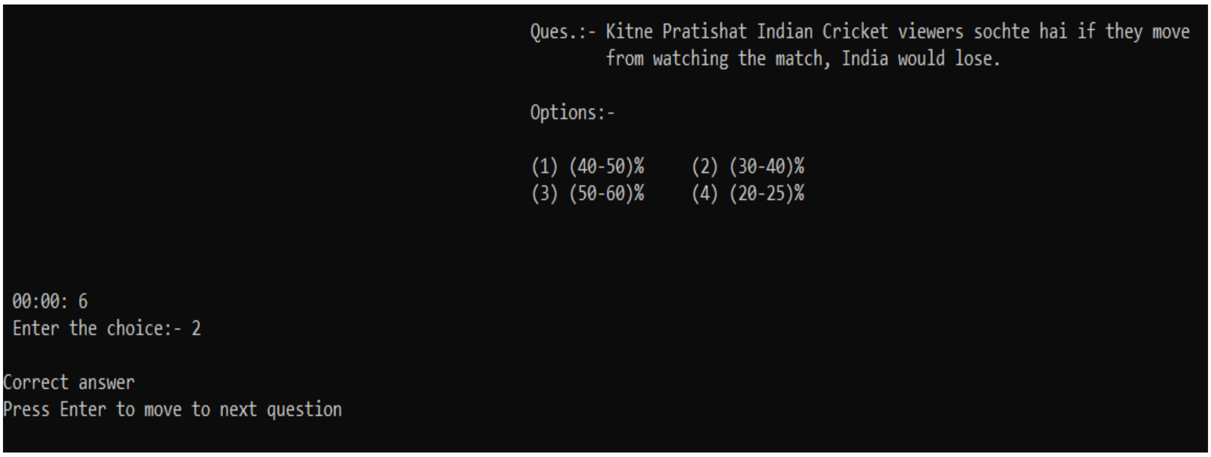
Pseudocode For Match The Following And Polling Round

```
Step 1: Import all header files.
Step 2: Declare the object function by:
            int match(*name)/int poll(*name);
Step 3: Declare main function
Step 4: Call the object function (object())/(poll()) in the main:
            print the welcome screen.
Step 5: Declare the random generator function.
int match(char* name)/int poll(char*name)
{
srand(time(NULL));
for(i=0;i<=n-1;i++)
{
temp=rand()%n; // n= no of question.
for(j=0;j<=i-1;j++){
if(temp==a[j])
break;}
if(j==i)
a[i]=temp;
else
i--;}
Step 6: Declare variable for the polling question
and start the timer.
Step 7: while(!kbhit() && TRUE)
        {
        printf("    \r 00:00:%2d",s);
                        ms++;
                        Sleep(10);
                        if(ms==41)
}
        system("cls");

Step 8: Display questions
Step 9: Declare time function //time limit will be according to the round

Step 10 : Accepts the user input and check:
            if(answer==right)
            marks=marks+1
            else
            marks=marks+0;
Step 11: After the completion of each round:
            if(total_marks>=required_score)
                enter to next round;
            else
                back to main menu;
```

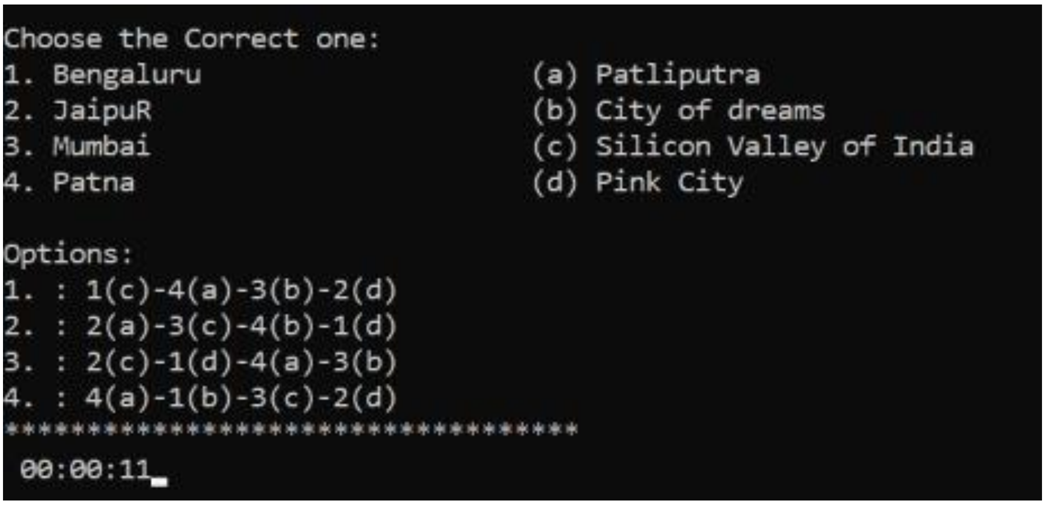
Output



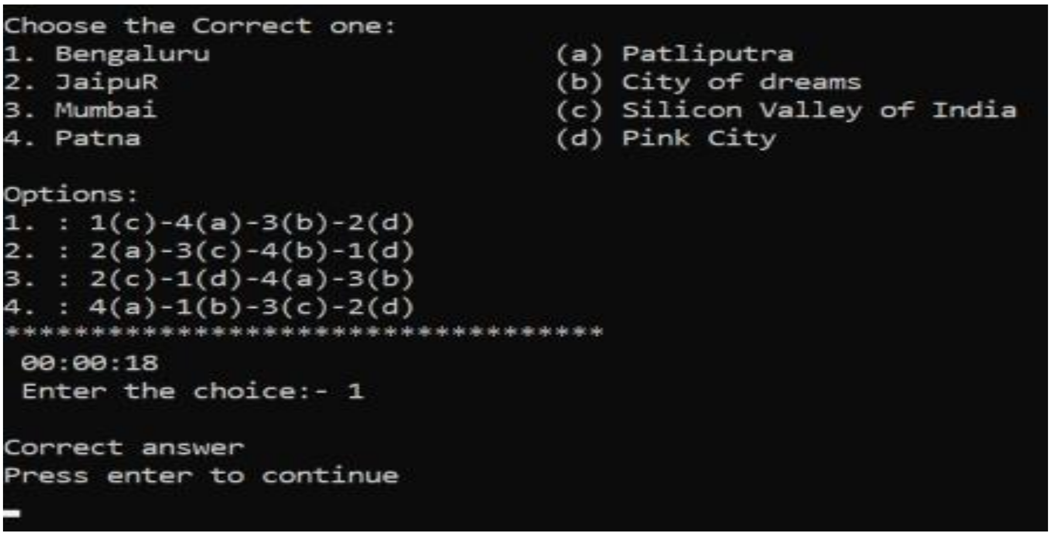
```
     *********************************
     ||     Lets start the game     ||

     || Entering to the basic round ||
     *********************************



  Press enter Key to continue the Game:- _
```



```
                                            Ques.:- Kitne Pratishat Indian Cricket viewers sochte hai if they move
                                                    from watching the match, India would lose.

                                            Options:-

                                            (1) (40-50)%     (2) (30-40)%
                                            (3) (50-60)%     (4) (20-25)%




 00:00: 6
 Enter the choice:- 2

Correct answer
Press Enter to move to next question
```



```
Choose the Correct one:
1. Bengaluru                      (a) Patliputra
2. JaipuR                         (b) City of dreams
3. Mumbai                         (c) Silicon Valley of India
4. Patna                          (d) Pink City

Options:
1. : 1(c)-4(a)-3(b)-2(d)
2. : 2(a)-3(c)-4(b)-1(d)
3. : 2(c)-1(d)-4(a)-3(b)
4. : 4(a)-1(b)-3(c)-2(d)
***********************************
 00:00:11_
```



```
Choose the Correct one:
1. Bengaluru                      (a) Patliputra
2. JaipuR                         (b) City of dreams
3. Mumbai                         (c) Silicon Valley of India
4. Patna                          (d) Pink City

Options:
1. : 1(c)-4(a)-3(b)-2(d)
2. : 2(a)-3(c)-4(b)-1(d)
3. : 2(c)-1(d)-4(a)-3(b)
4. : 4(a)-1(b)-3(c)-2(d)
***********************************
 00:00:18
 Enter the choice:- 1

Correct answer
Press enter to continue
_
```

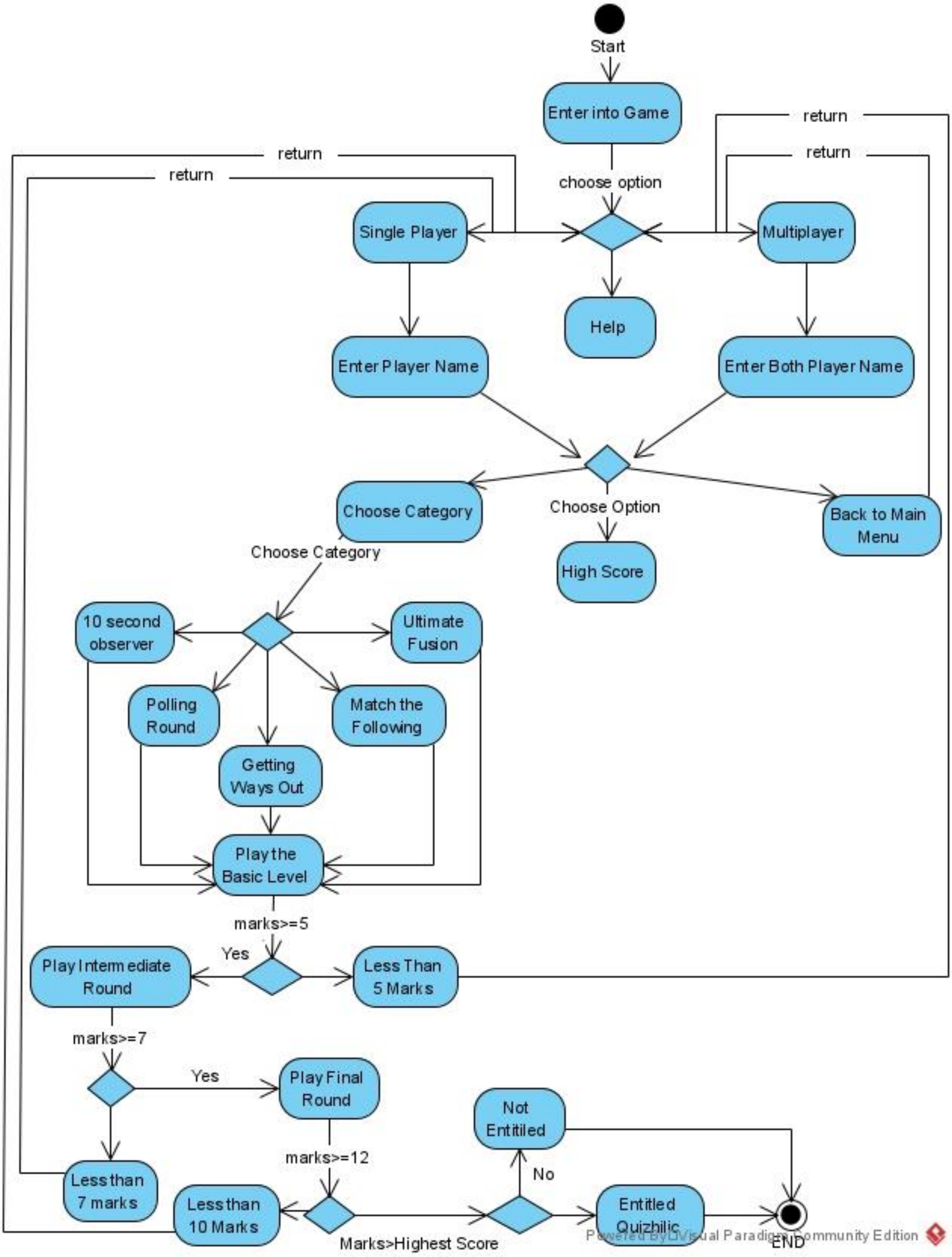
CONCEPT USED

1. We are including timer for each question. So, we are implementing this by using a header file windows.h and kbhit function.
2. We are using sleep() function in the timer to wait for the current thread for a specific time.
3. The questions will be displayed randomly, so we are using the random generator function rand().
4. For clearing screen after every question, we are using system(“cls”) function.
5. For displaying the name of the user/’s in uppercase we are using strupr() function along with the string.h header file.

CHALLENGES FACED

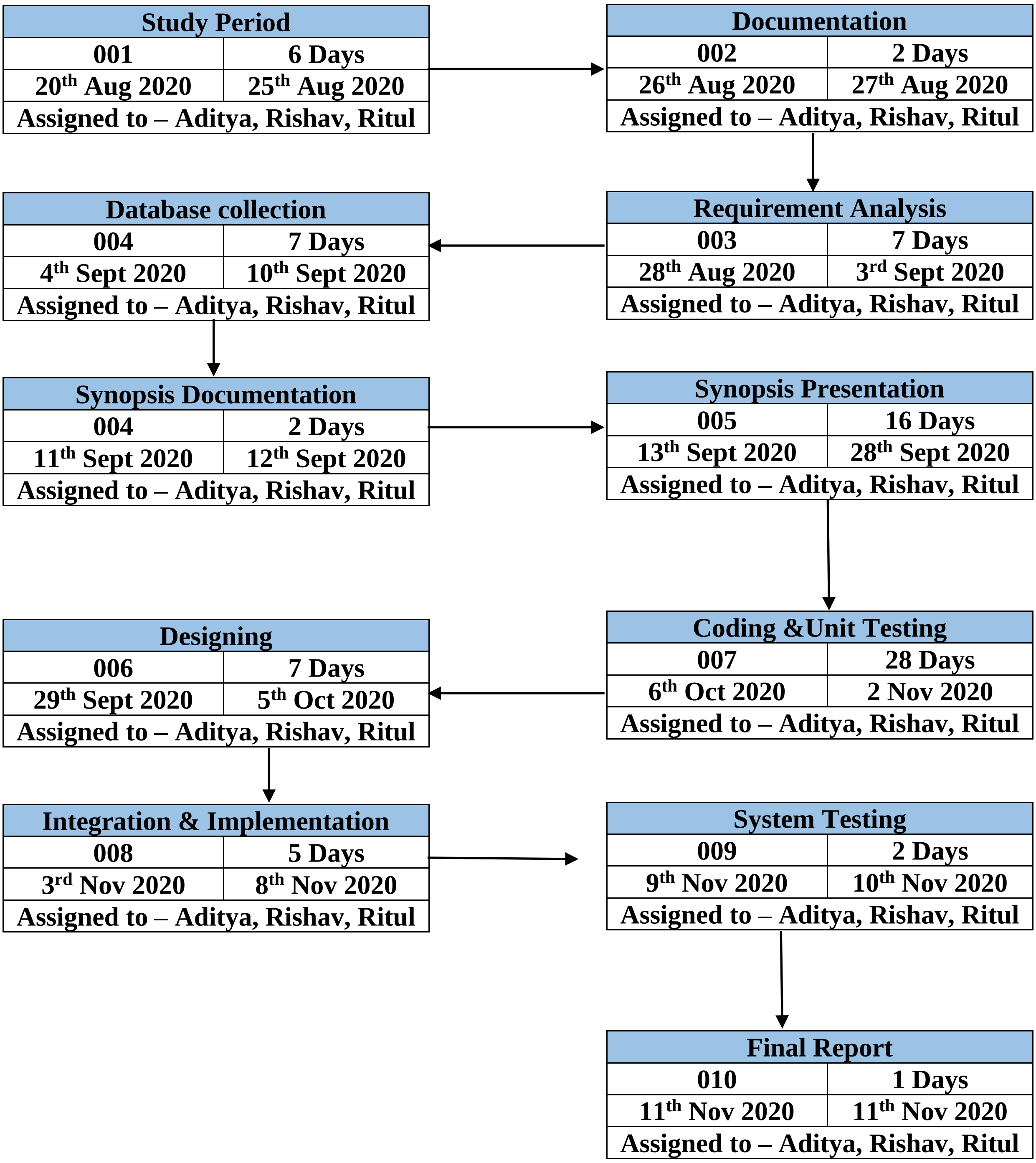
- Collection of unique questions for the game database of different categories.
- Implementation of timer for each question under different categories and inclusion of user input between the timer loop.
- As the timer loop was running, the control wasn’t going to the user input until the timer end.
- Using random generator for displaying the questions without repetition and assigning the range of the rand() function.
- For storing the high score so that the data wont get lost after closing the game.

Fuctioning of Game



PLAN OF WORK

Pert-chart of the schedule:



Study Period	
001	6 Days
20th Aug 2020	25th Aug 2020
Assigned to – Aditya, Rishav, Ritul	

Documentation	
002	2 Days
26th Aug 2020	27th Aug 2020
Assigned to – Aditya, Rishav, Ritul	

Requirement Analysis	
003	7 Days
28th Aug 2020	3rd Sept 2020
Assigned to – Aditya, Rishav, Ritul	

Database collection	
004	7 Days
4th Sept 2020	10th Sept 2020
Assigned to – Aditya, Rishav, Ritul	

Synopsis Documentation	
004	2 Days
11th Sept 2020	12th Sept 2020
Assigned to – Aditya, Rishav, Ritul	

Synopsis Presentation	
005	16 Days
13th Sept 2020	28th Sept 2020
Assigned to – Aditya, Rishav, Ritul	

Coding &Unit Testing	
007	28 Days
6th Oct 2020	2 Nov 2020
Assigned to – Aditya, Rishav, Ritul	

Designing	
006	7 Days
29th Sept 2020	5th Oct 2020
Assigned to – Aditya, Rishav, Ritul	

Integration & Implementation	
008	5 Days
3rd Nov 2020	8th Nov 2020
Assigned to – Aditya, Rishav, Ritul	

System Testing	
009	2 Days
9th Nov 2020	10th Nov 2020
Assigned to – Aditya, Rishav, Ritul	

Final Report	
010	1 Days
11th Nov 2020	11th Nov 2020
Assigned to – Aditya, Rishav, Ritul	

SYSTEM REQUIREMENT

Software Requirements

Name of Component	Specification
Operating System	Windows 10
Front End	C language
IDE Required	Dev C++

Hardware Requirements

Name of Component	Specification
Processor	Intel® Core™ i5-8250U CPU @ 1.60 GHZ 1.80 GH
RAM	4 GB
Hard Disk	1 TB

WORK DONE SO FAR

1. Documentation
2. Requirement Analysis
3. Collection of Database.
4. Designing of Main Window.
5. Study Analysis
6. Coding for Single-Player Categories

WORK TO BE DONE

1. Collection of databases for the questions.
2. Inclusion of timer for each question.
3. Implementation of Socket Programming
4. Implementing the algorithms under different rounds.
5. Integration of all the codes.
6. System Testing

REFERENCES

1. https://arxiv.org/pdf/1806.11525.pdf
2. https://arxiv.org/pdf/1812.01628.pdf
3. https://patentimages.storage.googleapis.com/38/6b/fe/25e0c281837bd6/US20090221372A1.pdf
4. https://pdfs.semanticscholar.org/7678/9b22b2862999b1a95cad0fc66d21e783012d.pdf
5. https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/#:~:text=The%20iterative%20waterfall%20model%20provides,from%20the%20classical%20waterfall%20model.&text=When%20errors%20are%20detected%20at,by%20programmers%20during%20some%20phase.
6. https://www.geeksforgeeks.org/count-number-ways-reach-destination-maze/#:~:text=Given%20a%20maze%20with%20obstacles,%2C%20j%2B1)%20only.
7. https://patents.google.com/patent/US20090221372A1/en
8. https://www.google.com/url?sa=i&url=https%3A%2F%2Fmedium.com%2F%40bhaktithaker29.bt%2Frabin-karp-the-string-matching-algorithm-ca0ba1f8e5fa&psig=AOvVaw3w63-REJPiF_FeMAGN0Pky&ust=1600849011224000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCOCq6Lip_OsCFQAAAAAdAAAAABAD
9. https://www.professionalqa.com/
10. https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/