**VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM 590014**

****

Report on

**“HANGMAN”**

By

Sharan Rudresh(1BM16CS092)

Samrudh H N (1BM16CS086)

Under the Guidance of

**Pallavi G B**

Assistant Professor, Department of CSE

BMS College of Engineering

Work carried out at

****

Department of Computer Science and Engineering

BMS College of Engineering

(Autonomous college under VTU)

P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

2017-2018

**BMS College of EngineerinG**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

***CERTIFICATE***

This is to certify that the assignment titled “Hangman” has been carried out by Sharan Rudresh(1BM16CS092), Samrudh H N (1BM16CS086), during the academic year 2017-2018.

Signature of the guide Signature of Examiner

**Pallavi G B**

Assistant Professor

Department of Computer Science and Engineering

BMS College of Engineering, Bangalore

**BMS College of EngineerinG**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

****

***DECLARATION***

We, Sharan Rudresh (1BM16CS092), Samrudh H N (1BM16CS086), students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this assignment work entitled "HANGMAN" has been carried out by us under the guidance of

**Pallavi G B,** Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Aug-Dec 2017. We also declare that to the best of our knowledge and belief, the assignment reported here is not from part of any other report by any other students.

**Name**  **Signature**

Sharan Rudresh (1BM16CS092)

Samrudh H N (1BM16CS086)

**DESCRIPTION OF EXISTING MODULES:**

**1.Introduction:**

Hangman is a guessing game, in which the player has to guess the word or words. Initially the player is shown a series of blank spaces. The player has to guess the word letter by letter. The player has only 10 guesses. The player has to guess the word before running out of guesses. Points will be deducted for every wrong guess made.

**2. Functionalities of the Existing Code:**

* The program does not make use of any classes.
* Three character arrays are used. The first array stores the word that the player has to guess. The second array is the series of blanks displayed initially to the player. This is array is updated every time the player guesses a correct letter. The third array stores all the guesses made by the player, to ensure that the player doesn’t make the same guess twice.
* The number of guesses given to the player depends on the length of the word.
* The player may choose to play against the computer, in which case the word is chosen randomly from a text file in the system.
* The player has an additional option to add words to the text file.
* If the player chooses to play against another player, then the word is given by one of the players.
* Each word stored in the text file belongs to one of the categories- Movie, Song, Animal or Sport. The category of the word is identified by the character preceding the word. When reading from the file, the first character is passed to a function which displays the category to the player.

**DESCRIPTION OF MODULES ENHANCING THE EXISTING CODE**

1. **List of new modules designed**

* Word class  
   Stores all information related to the word.
* Game class, inherits from word.

Stores all information related to the game

* Classes Level1 & Level2 inherited from Game.
* Class PvP - Player vs Player, inherits from Game.

**The Word Class:**

The word class contains three character arrays to store the *word*, *dashes* and*letters*, to store the unique characters in the word. There’s one character to store the *category*. Two integers, to store the *length* of the word and the number of digits.

*The makedash()* function is used to initialize the *dashes* array in the code. This function checks for individual characters from the word and converts only the letters into dashes while the spaces and punctuations are not changed.

The *type\_of\_word()* function prints the category of the word. There are many different categories of words in the text file and to make things easier this code checks for the category of the word prints it.

The *reveal\_one\_alpha(char[])* function reveals a random letter to the player. If the player enters ‘+’, then a random letter is revealed to the user. This function can be used only once and the player loses 5 points.

*Check(char)* function takes a letter from the user and checks if this letter is present in the word. If the letter is present in the word, it replaces the corresponding dashes with the letter and it returns 1.

The function *let\_in\_word()* initializes the letters array with the unique characters in the word.

*Group(char x[])* function checks if the character array x[] matches any part of the word.

*Show()* function displays the dashes.

*Theword()* displays the word and capitalizes the first letter of each word.

*Compare()* compares word and dashes, returns 1 if they are the same else it returns 0.

**The Game Class:**

The game class contains three character arrays- chosen[], incorrect[] and level[].

* Chosen[]: Stores the characters guessed by the player.
* Incorrect[]: Stores the incorrect guesses that the player enters.
* Level[]: Stores the levels-level1 & level2.

There are variables to store the number of wrong guesses, score, turns left and the time taken.

The function *draw()* is used to draw the hangman image. One part of the image is displayed every time the player enters a wrong guess. This function uses switch statement to display the image after every wrong guess.

The function *play()*: This is where the player guesses the word by entering one letter at a time. The player can get a hint of one random letter in the word by entering ‘+’. The player can also enter ‘-‘ to guess the entire word at once and the player receives a bonus of 10 points of the entered word is correct but if it is wrong, the player’s points is reduced by 5. The player can make only 10 mistakes and for every mistake i.e. for every wrong guess the player loses 1 point.

The *comp\_chooses()* function is executed if the player wishes to play against the computer. The computer selects a random word from the text files.

The virtual function *player\_adds()* is used if the player wishes to add words to the file.

The virtual function *scoring()* is used to calculate the score based on the number of wrong guesses and the time taken by the player to guess the full word.

The virtual function *player\_chooses()* is used when the player wants to play against another player. The first player is made to enter a word. This word is converted into dashes and the second player guesses the word. The scoring system is similar to when playing against the computer.

**Classes Level1 & Level2:**

The default constructor initializes the score to 20 in class *Level1*. For level2 it initializes score to 30.It initializes character array *level*of the parent class *game* to the name of the corresponding file – *level1.txt* or *level2.txt*.

The virtual functions *player\_adds()* and *scoring()*of the parent class *game* are defined inside both these classes.

**PvP Class – Player vs Player:**

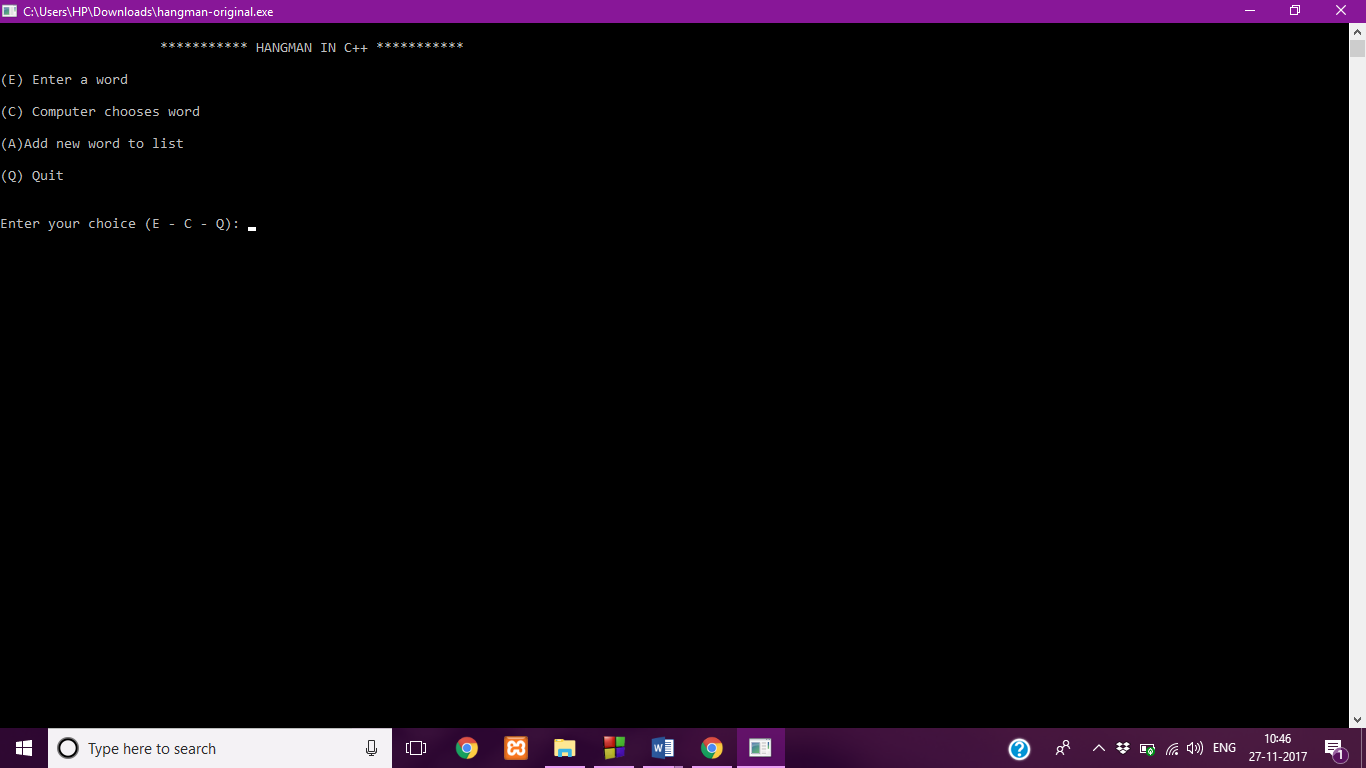
Thevirtual function *player\_chooses()* of the parent class *game,* is defined inside this class. This function is called when the player chooses to play against another player. The first player enters a word. This word is converted into dashes and the second player guesses the word. The scoring system is similar to when playing against the computer.

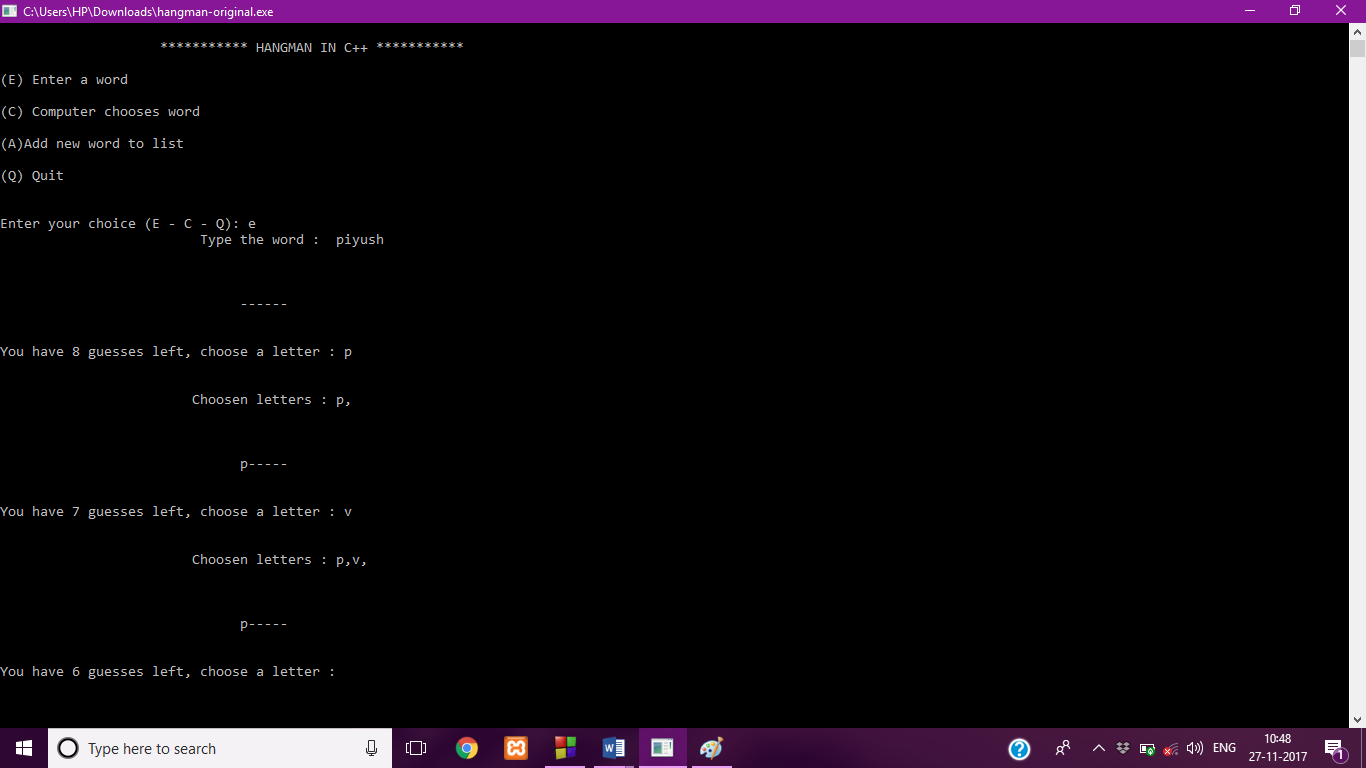
**Mechanisms used from C++:**

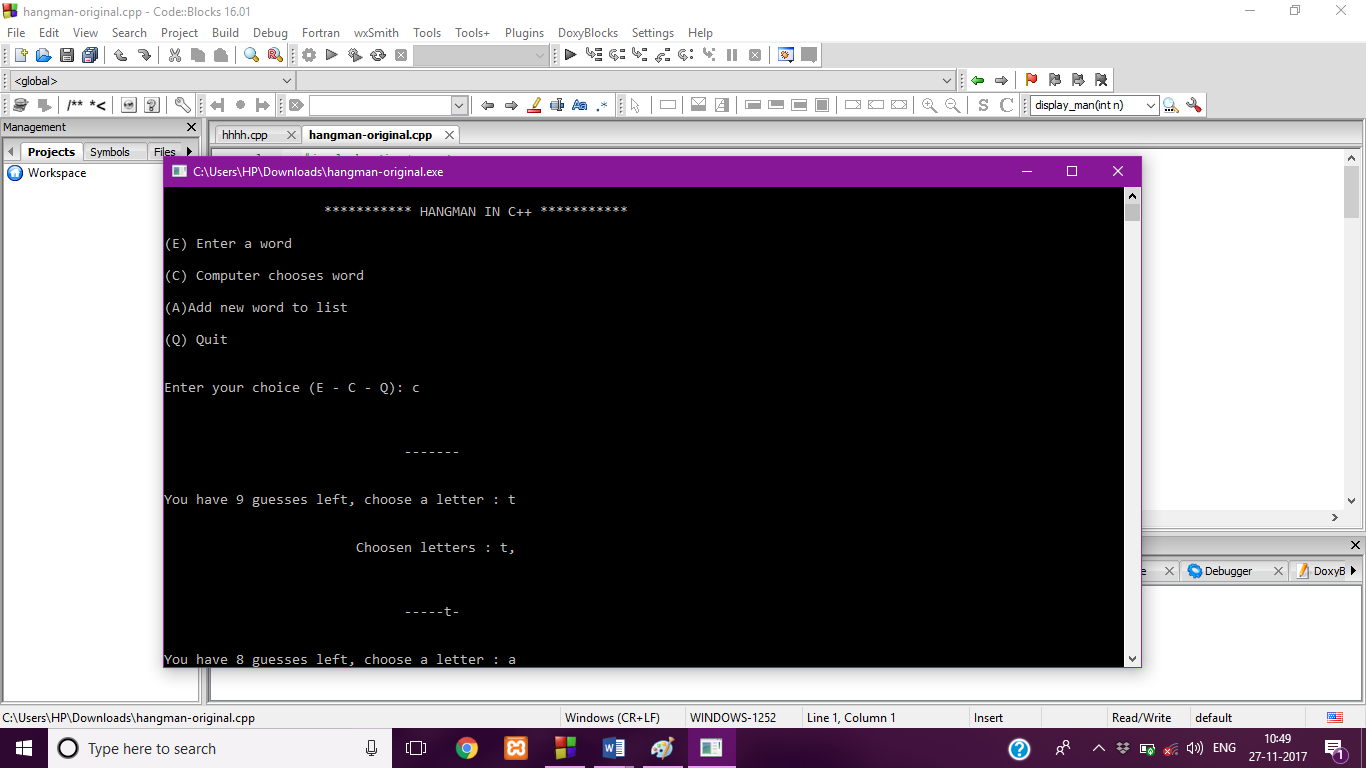
* File Handling  
   The words are stored in two text files, one for each level. Words are chosen at random from the text file.
* Multi-Level Inheritance  
   The *Word* class is inherited by the *game* class. Classes *Level1, Level2* and *PvP* are derived from the *game* class.
* Virtual Functions  
  The game class has virtual functions, which have separate definitions in the child classes.
* Polymorphism  
   A pointer to the *game* class is declared. Based on the player’s choice, an object of the appropriate class is dynamically created.

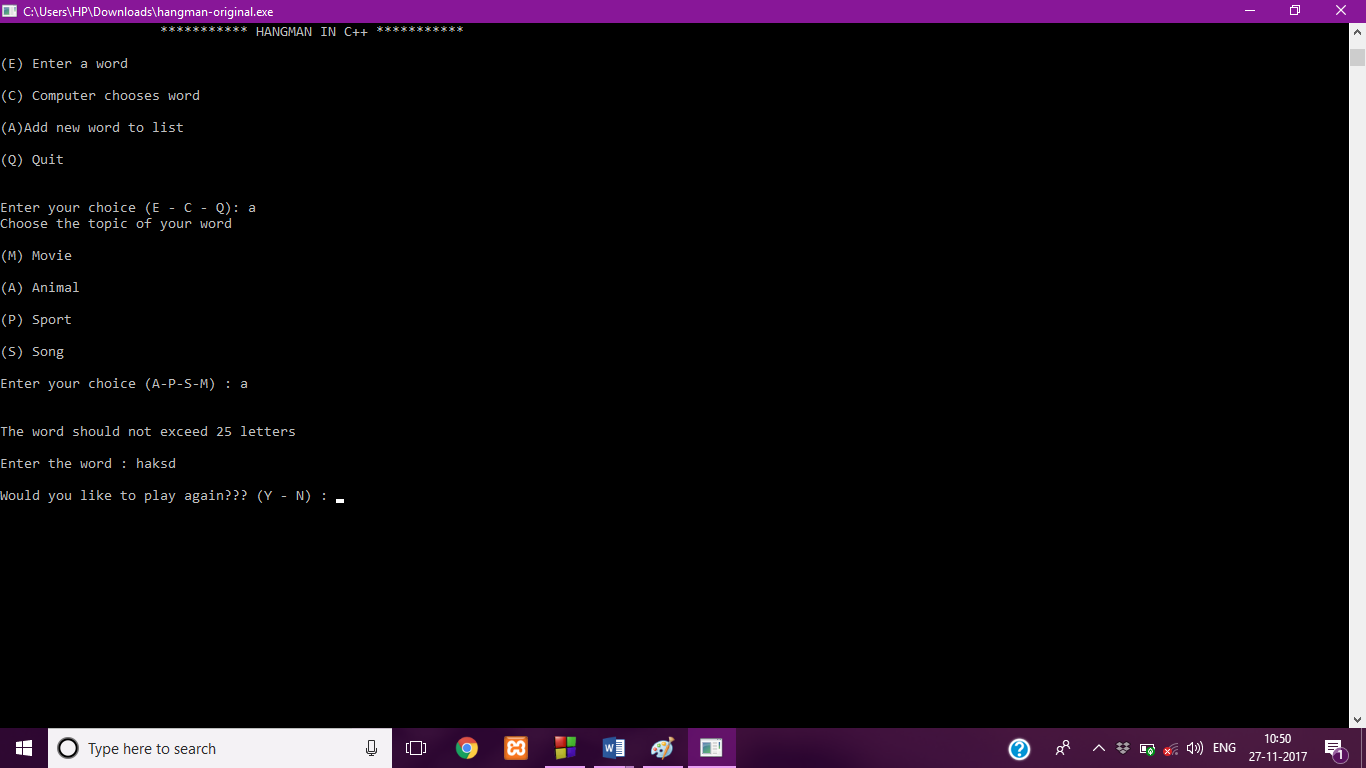
**SCREENSHOTS:**

* **EXISTING MODULES**



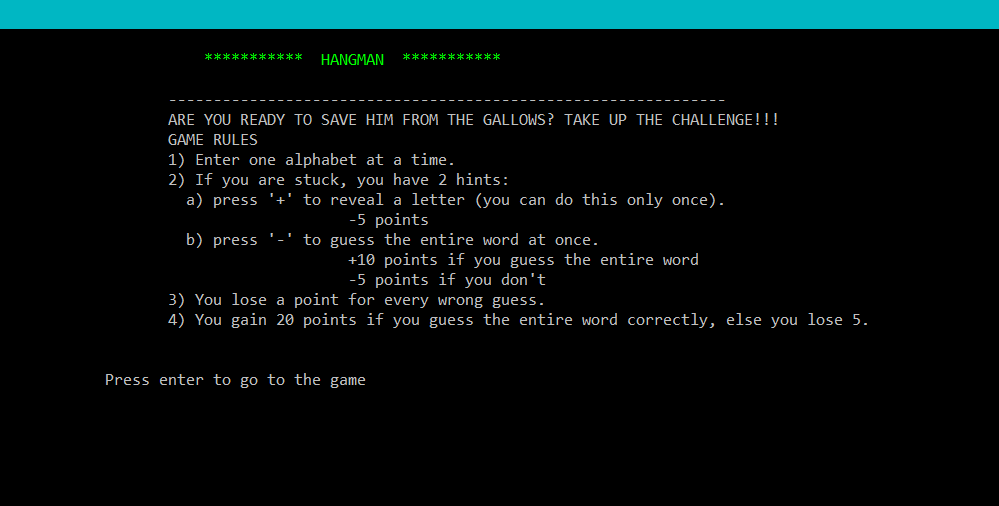


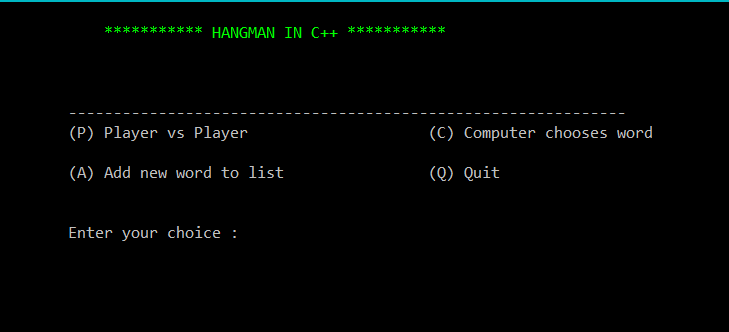




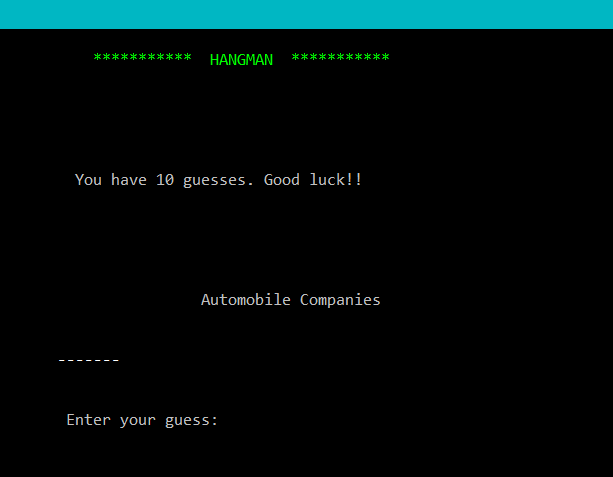
**NEW MODULES DESIGNED**

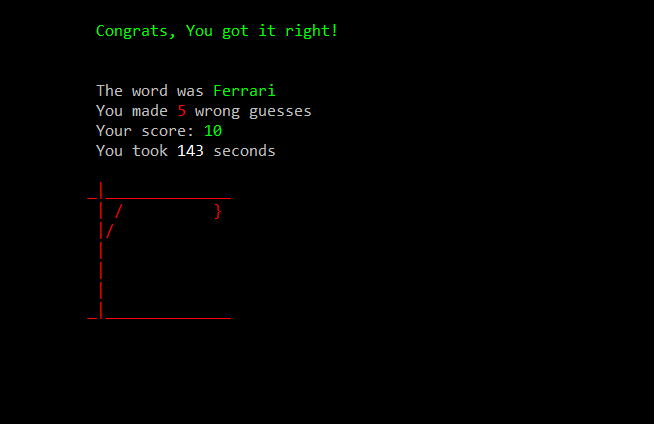
Starts with the rules

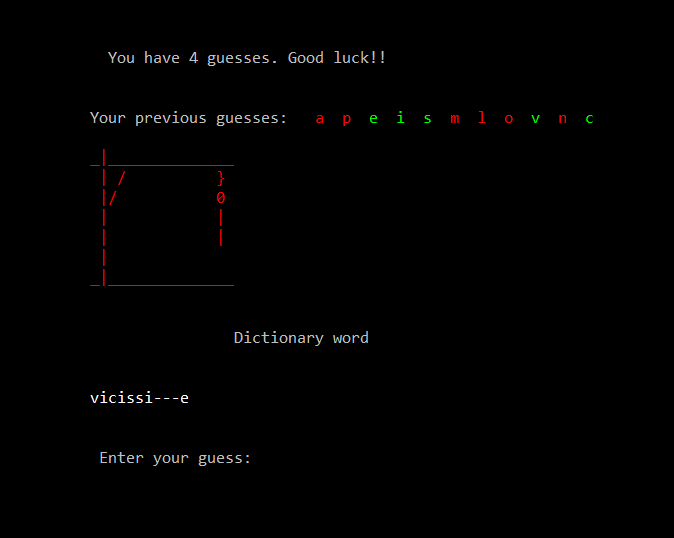




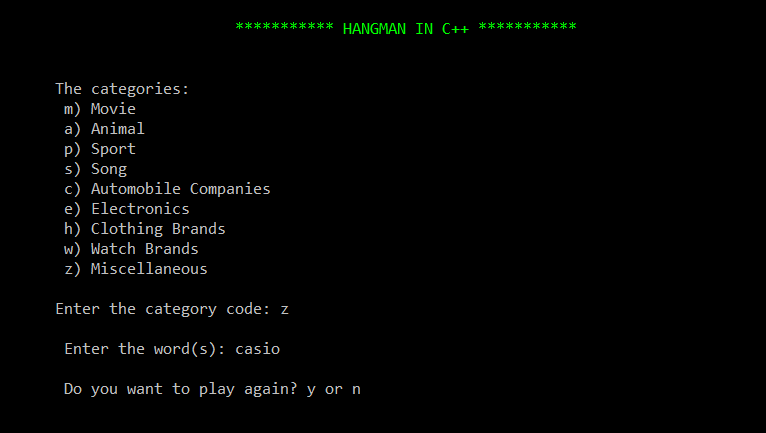
Against computer



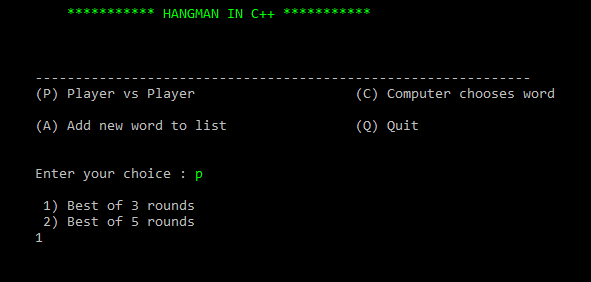


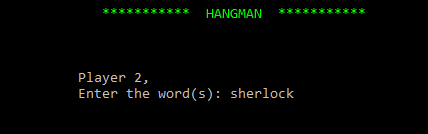


**Adding words to the file**

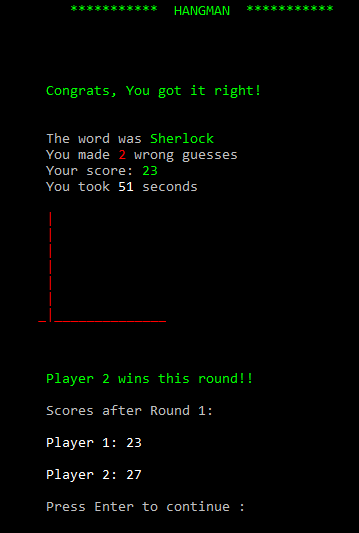


**Player vs Player**

****

****

****

****

**OUTCOMES OF THE ASSIGNMENT WORK**

* Used files to store the words. We learnt the different ways of reading a file and writing into a file.
* Learnt the different types of inheritance and also the different modes of inheritance.
* Learnt to use the time() function to calculate the time difference and also to generate a random number.
* Learnt how to use dynamic polymorphism.
* Used Windows.h functions to change the colour of the output.