**CHAPTER 1:**

**INTRODUCTION**

The Hangman Game is a program that is created for the purpose of entertainment and increasing the social media presence. It is a fully computerized game which keeps up with interest of the today’s generation.

* 1. **Why our topic is important**

We live in the digital age. Hence the next generation also expects everything to be digitalized and easier. People of all age are dependent on Technology. And as a matter of fact the computer games are just as popular with the kids as well as the adults. In today’s world the world of 1’s and 0’s are more interesting and captivating and it has higher selling grounds than the rest of them. Over the past 45 years, gaming in all forms has shown remarkable growth and development across the globe. It is now everywhere in our lives, from video games on our consoles, PCs and mobiles to the “gaming” that has become part of our social media, work-life and messaging apps. The relationships between particular game genres and things like pattern recognition, critical thinking, time management and resource efficiency are explored. A lot of the gamification of the mainstream internet such as reputation and rankings, messaging, and even likes and shares also have roots in video games. This kind of crossover of gaming principles and technology into other areas of our lives should not be overlooked.

* 1. **Where is it used? Applications**

The aim of this project was to develop a light-weight game coding for everyone to enjoy. Therefore we have decided to bring the old school game HANGMAN to the reality of digital age. We wouldn’t let our old school memories die with time, so we put our minds together to create a game which can be enjoyed by people of all ages.

**1.3 What is the game about?**

We used the arrays data structures fir this project. This is a two player game where player 1 will enter the name of the movie for his friend to guess. Thus the game begins.

Then the system is handed off to the second player where the actual game starts. The second player is asked to start guessing the name of the movie letter by letter. If the letter is guessed wrong the body parts of the man starts to draw. Since the number of tries the player 2 gets is 6, there are 6 body parts to be drawn; those are the head, the abdomen, the left and right arm and the left and right leg.

Once the picture of the man is complete (the number of tries is 0), the man is hung and the second player has lost the game.

**1.4 Overview**

In the next part of this report we will see what we have actually done in this project, how we have done it and most importantly why we have done it, along with the proof of outcomes and codes and snippets of the project and how it can be improvised later.

**CHAPTER 2:**

**PROBLEM STATEMENT**

The aim of this project was to develop a light-weight game coding for everyone to enjoy. Therefore we have decided to bring the old school game HANGMAN to the reality of digital age. We wouldn’t let our old school memories die with time, so we put our minds together to create a game which can be enjoyed by people of all ages, So here are the problems that were faced:-

## **2.1 PROJECT ESTIMATION**

Estimating any type of project can be hard, but it is especially true for video games. For an industry that is as fast changing as this, you have to keep a tight lid on schedule. Otherwise you’ll end up always iterating and implementing new ideas until the end of time.   
A cool new feature always sounds good, but do you have the time? Does the benefits outweigh the cost? Does much of the code have to be changed?  
These are just some of the questions you have to answer before starting anything new.  
There are countless development teams who failed to deliver because of poor time allocation. Having good management is key, someone who oversees every aspect of the development process and can evaluate the ideas and shut them down or postpone them to another project.

**2.2 POLISH**

Games are carefully designed experiences. And nothing can ruin this more like a badly executed feature, weird anomalies and bugs. Some say, that the last 10% of development time is more important than the first 90%, and it’s true to some extent. It can be really demanding to polish everything in time, when you are facing a tough release schedule.  
The best way to avoid long crunch periods is to keep a tight lid on executing everything don’t just put them off for later. The closer the deadline is, the more of the things you put off will come back to haunt you.

**CHAPTER 3:**

**OBJECTIVES**

As life gets increasingly busy, it becomes more difficult for people to spend time with their loved ones. Thus, it is necessary to keep everyone entertained to ensure a relaxing time, so the objective of our project is to make sure that entertainment and skill enhancement both are balanced.

**1. Benefits to be provided by the game**

It improves your skill set and your focusing power also makes sure that you mind relaxes.

**2. To create a fully computerized game**

Since today’s generation is built on computers and software’s, we intend to lean towards that to make it more captivating.

**3. Enhances the capability to decode sequences**

The more the game is played the better a person gets at mental logic solving.

**4. Great selling point**

This game is light weight which can run on any platform, this makes it easier to play and handle.

**5. Creates an interface between the players and the system**

Interface is another very important aspect of game programming. The interface is the mode of communication between the computer and the player. Like any human language, it is the funnel through which the programmer must squeeze the avalanche of thoughts, ideas and feelings that she seeks to share with the fellow player. Interface will dictate what can or cannot be done.

**CHAPTER 4:**

**SYSTEM REQUIREMENTS**

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behaviour of a system or software application. It includes a variety of elements that attempts to define the intended functionality required by the customer to satisfy their different users.

In addition to specifying how the system should behave, the specification also defines at a high-level the main business processes that will be supported, what simplifying assumptions have been made and what key performance parameters will need to be met by the system.

**4.1 Software Requirements**

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client’s point of view.

It defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The software requirements in our program are

1. Programming language: C.
2. Compiler: Online GCC compiler
3. Operating System: Windows.

**4.2 Hardware Requirements**

There are hardware requirements, also known as system requirements, for every OS we are going to use. These requirements include the minimum processor speed, memory, and disk space required to install Windows. In almost all cases, you will want to make sure that your hardware exceeds these requirements to provide adequate performance for the services and applications running on the server. The table below outlines the minimum hardware requirements to execute this project.

1. Processor: Any 3rd Gen+ Intel Processor or even a 1st gen AMD processor.

2. RAM: Since here it was done on VMware Workstation 15, it might require minimum of 4GB of RAM if it was just done on Ubuntu then 2GB of RAM.

3. Graphics: Minimal.

**4.3 Why we have used Windows**

Since our aim was to create a user friendly and light weight game, we have used the most widely used software i.e. Windows. This game also be played using Ubuntu’s terminal. The compiler compiles the code and the then run it. This code works on all the platforms.

**CHAPTER 5:**

**METHODOLOGY**

An array is a collection of one or more values of the same type. Each value is called an element of the array. The elements of the array share the same variable name but each element has its own unique index number (also known as a subscript). An array can be of any type, For example: int, float, char etc. If an array is of type int then its elements must be of type int only.

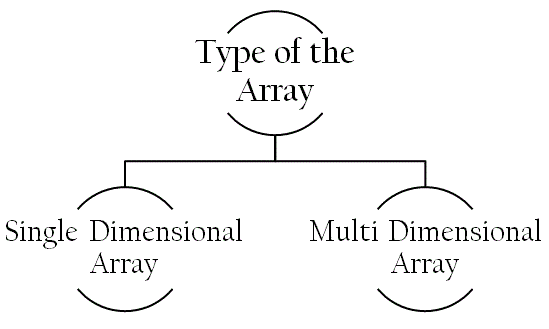


Fig 5.1 Types of array

There are two types of arrays i.e. Single-Dimensional Arrays and Multi-Dimensional arrays. Conceptually you can think of a one-dimensional array as a row, where elements are stored one after another.

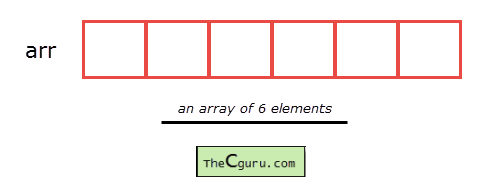


Fig 5.2 Memory allocation of array

A multi-dimensional array is an array that has more than one dimension. It is an array of arrays; an array that has multiple levels. The simplest multi-dimensional array is the 2D array, or two-dimensional array. It's technically an array of arrays, as you will see in the code. A 2D array is also called a matrix, or a table of rows and columns.



Fig 5.3 Memory allocation of multi-dimensional array

In our code we use arrays to store the word given by player 1 also to display the output and to store the guessed letters temporarily in array. We use loops to determine if the letters guessed are correct or wrong.

**5.1 FLOW CHART**

This is the flow chart of the working of the game.

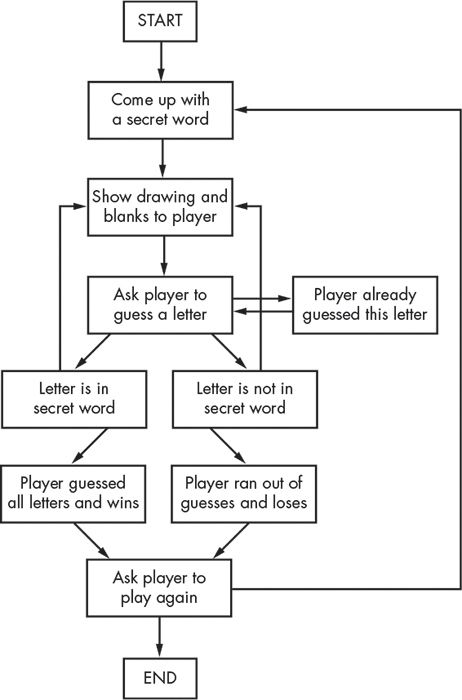


Fig 5.4 Flow chart of working of game

**CHAPTER 6:**

**CODE IMPLEMENTATION**

In this section we shall explain the working of the code. We shall divide the code in different segments to make it easier to explain the code.

**SEGMENT 1:**

This segment includes the pre-processor directives and the function declaration.

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

void showHangman(int);

**SEGMENT 2:**

This segment contains the first part of the main function. Here we declare the arrays and the variables that we need to use. Also we use print statements to interact with the players.

{

char hangmanWord[100], tempWord[100];

char hangmanOutput[100];

int wrongTry = 6 , matchFound = 0;

int counter = 0 , position = 0, winner, length , i;

char alphabetFromUser;

system("cls");

printf("\n\n PLAYER 1 : ENTER THE MOVIE NAME FOR YOUR OPPONENT TO GUESS!");

printf("\n\n\t Enter HERE ==> ");

scanf("%s",hangmanWord);

printf("\n\n Now give the COMPUTER to your friend and see if he can CRACK it!!!");

length = strlen(hangmanWord);

printf("\n\n !!!!!!!!!!!!!!!!!!!Welcome to the HANGMAN GAME!!!!!!!!!!!!!!!!!\n\n\n");

printf("\n\n You will get 6 chances to guess the right word");

printf("\n\n So help the Man and get...set...GO..!!");

getchar();

system("cls");

printf("\n\t||===== ");

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

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

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

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

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

**SEGMENT 3:**

This is the part where the number of the letters is revealed to the second player. Also the layout of the game is displayed.

printf("\n\n The word has %d alphabets \n\n",length);

for( i = 0; i < length ; i++)

{

hangmanOutput[i] = '\_';

han gmanOutput[length] = '\0';

}

for(i = 0 ; i < length ; i++)

{

printf(" ");

printf("%c",hangmanOutput[i]);

}

**SEGMENT 4:**

This the while loop segment which is the major part of the entire gaming code.

while(wrongTry != 0)

* While loop for exiting the program when no try left

{

matchFound = 0;

printf("\n\n enter any alphabet from a to z and please use small case!!");

printf("\n\n\t Enter HERE ==> ");

fflush(stdin);

scanf("%c",&alphabetFromUser);

if(alphabetFromUser < 'a' || alphabetFromUser > 'z')

* In case player gives input other than 'a' to 'z' the console will ask again

{

system("cls");

printf("\n\n\t Wrong input TRY AGAIN ");

matchFound = 2;

}

fflush(stdin);

if (matchFound != 2)

{

for(counter=0;counter<length;counter++)

* For loop to check whether player input alphabet exists or not in the word

{

if(alphabetFromUser==hangmanWord[counter])

{

matchFound = 1;

}

}

if(matchFound == 0)

* In case of wrong guess

{

printf("\n\t :( You have %d tries left ",--wrongTry);

getchar();

showHangman(wrongTry);

getchar();

}

else

{

for(counter = 0; counter < length; counter++)

{

matchFound = 0;

if(alphabetFromUser == hangmanWord[counter])

{

position = counter ;

matchFound = 1;

}

if(matchFound == 1)

{

for(i = 0 ; i < length ; i++)

{

if( i == position)

{

hangmanOutput[i] = alphabetFromUser;

* Put the alphabet at right position

}

else if( hangmanOutput[i] >= 'a' && hangmanOutput[i] <= 'z' )

* If the position already occupied by same alphabet then no need to fill again EASY!! and continue

{

continue;

}

else

{

hangmanOutput[i] = '\_';

* Put a blank at not guessed alphabet position

}

}

tempWord[position] = alphabetFromUser;

* Put the alphabet in another char array to check with the original word

tempWord[length] = '\0';

* Put the NULL character at the end of the temp string

winner = strcmp(tempWord,hangmanWord);

* Upon True comparison it will return 0

if(winner == 0)

* If the player guessed the whole word right then he/she is the WINNER

{

printf("\n\n\t \t YAHOO!!!!! You are the WINNER !!!!!");

printf("\n\n\t The Word was %s ",hangmanWord);

printf("\n\n\n\n\t\tEASY HUH???\n\n");

getchar();

return 0;

}

}

}

}

}

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

for(i = 0 ; i < length ; i++)

{

printf(" ");

printf("%c",hangmanOutput[i]);

* Show the original Word With blanks and right Input alphabet

}

getchar();

}//end of while loop

**SEGMENT 5:**

This is the switch case for the visuals of the game. Each time the player guesses a wrong letter there is one part of the body drawn. Here we aim that the man is not completely drawn otherwise the game is OVER.

void showHangman(int choice)

{

switch(choice)

{

case 0:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| %cO/",'\\');

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

printf("\n\t|| / %c",'\\');

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

break;

case 1:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| %cO/",'\\');

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

printf("\n\t|| %c",'\\');

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

break;

case 2:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| %cO/",'\\');

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

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

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

break;

case 3:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| %cO/",'\\');

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

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

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

break;

case 4:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| %cO ",'\\');

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

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

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

break;

case 5:

system("cls");

printf("\n\t||===== ");

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

printf("\n\t|| O ");

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

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

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

break;

}

return;

}

**CHAPTER 7:**

**RESULT WITH THE OUTPUT**

**CASE 1:**

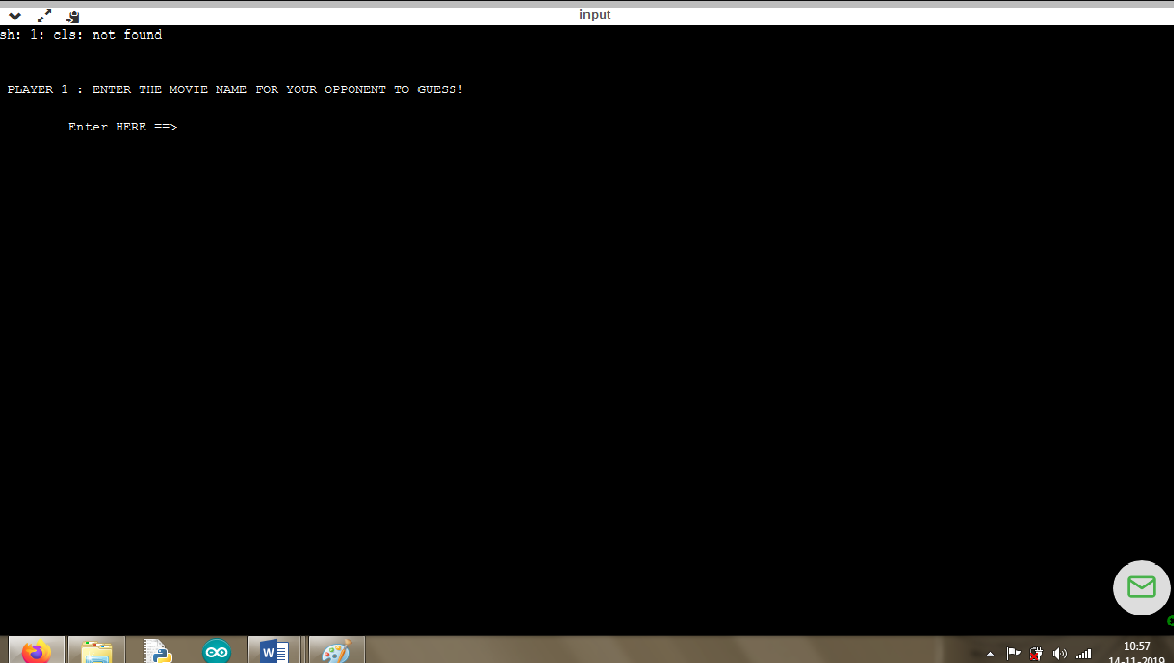


Fig 7.1 Output for first player

This is the first part of the output which introduces the players to the game. Here the player 1 is supposed to give a movie name to the computer.

**CASE 2:**

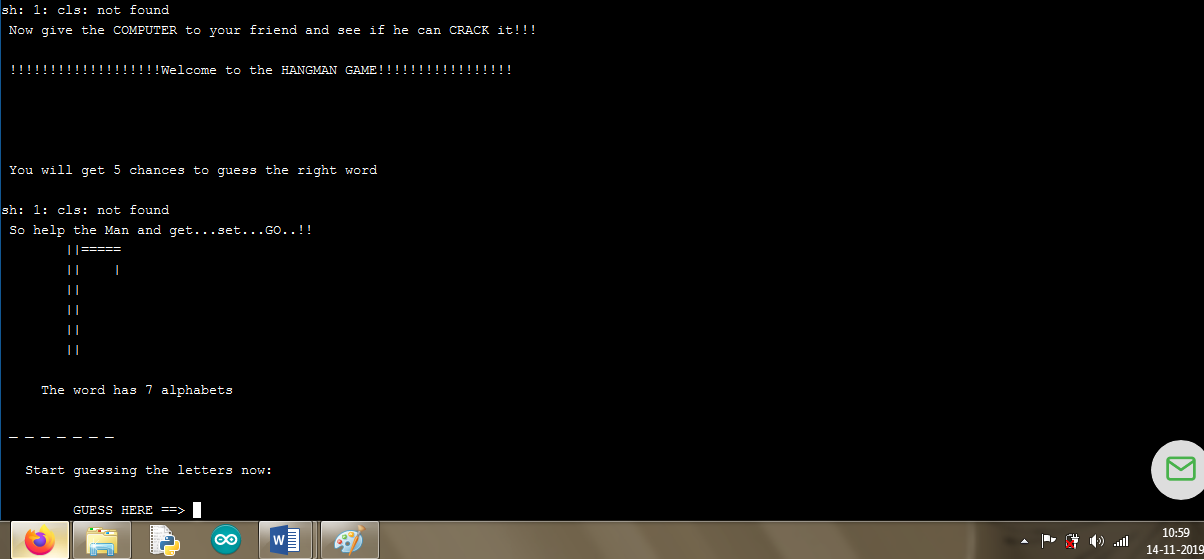


Fig 7.2 Output for second player

This is the second part of the output window where the player 2 is supposed to handle the system. The second player has to start guessing the letters one by one.

**CASE 3:**

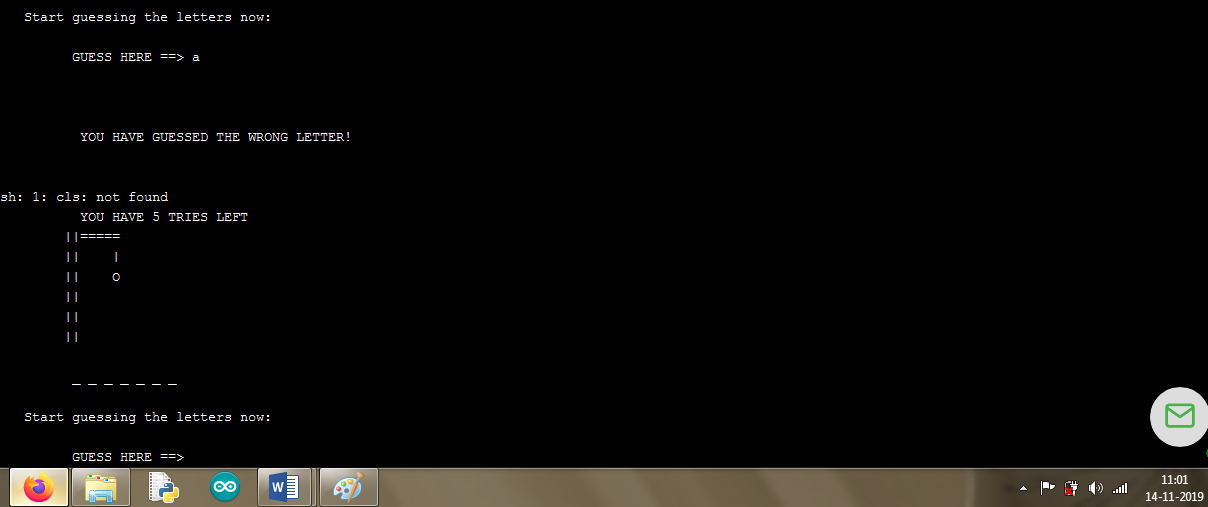


Fig 7.3 Output after first letter entered

If the letter guessed by the player is correct then the letter is shown in the output while other letters are still missing.

**CASE 4:**

If the letter guessed by the player is wrong then the first body part of the man is drawn and displayed. Then the number of tries are then displayed. This process keeps repeating until you exhaust all you tries.



Fig 7.4 Output with first wrong guess

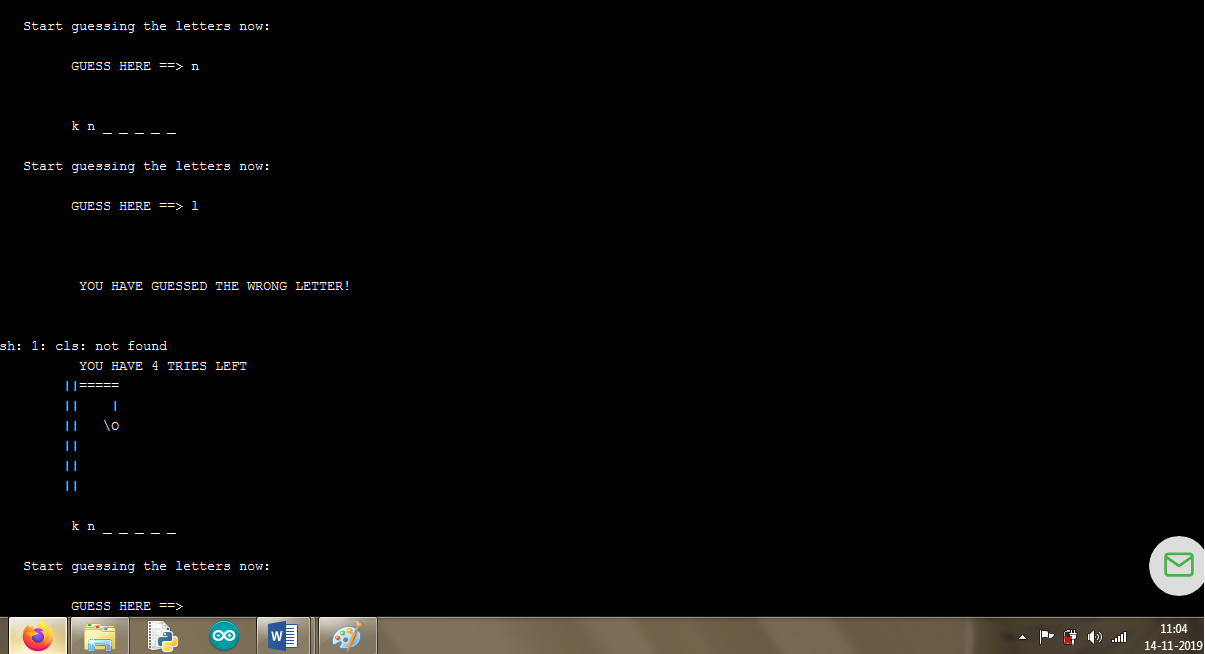


Fig 7.5 Output with one correct guess and one wrong guess

As the number of wrong tries keep increasing more the body parts are drawn. When all your tries are over you lose. The man is hung.

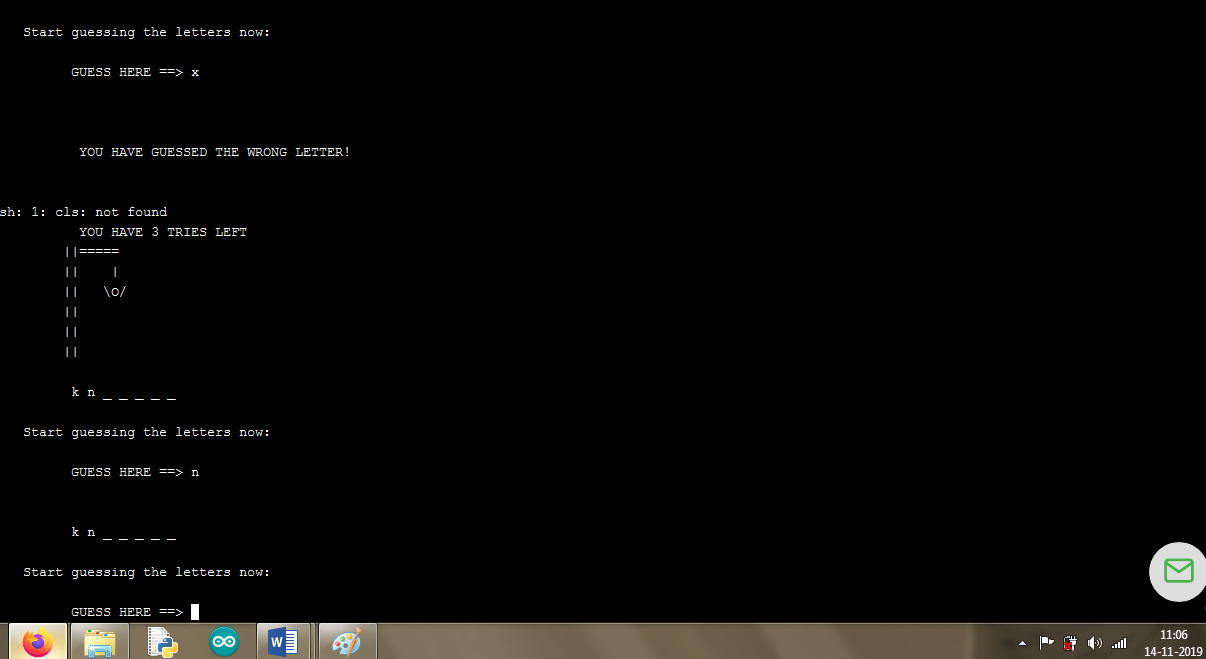


Fig 7.6 Output with third wrong guess

When we enter the correct letters then the word is displayed with the letters that you entered. This way you can continue on with the game. If the letter you entered was wrong then the body part of the man is displayed as well as the number of tries that is left.

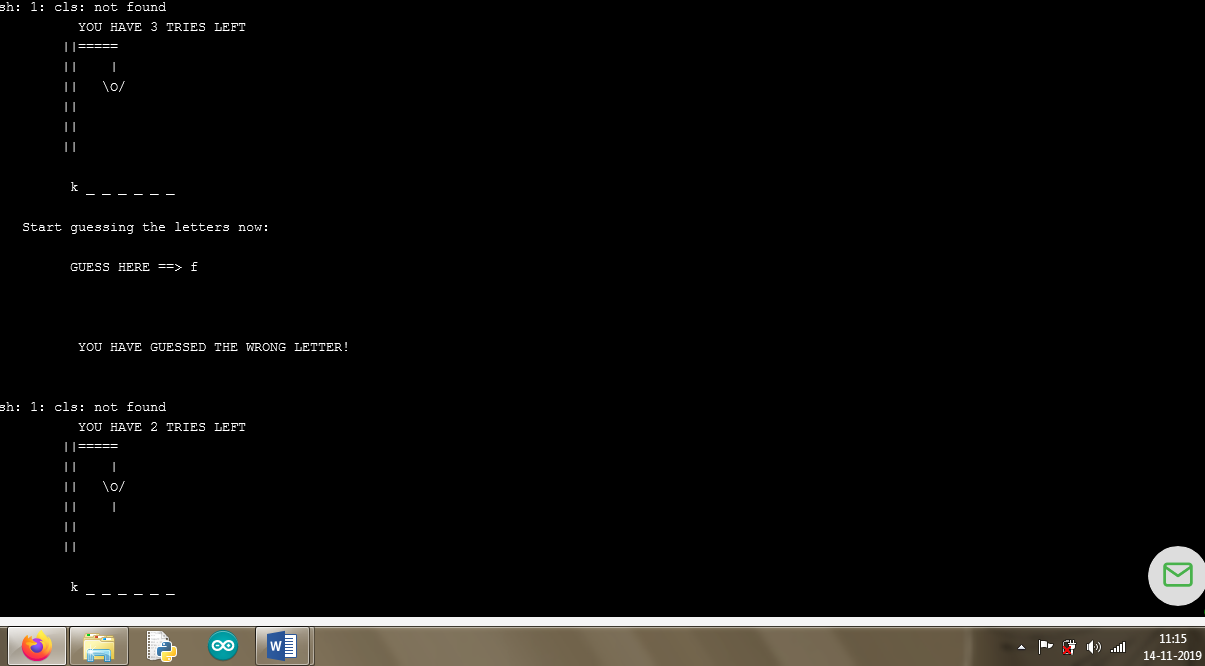


Fig 7.7 Output with fourth wrong guess

Here as you can see the letter we entered was wrong which means the body part of the man was drawn. Once you have completely guessed all the letters that means you have lost.

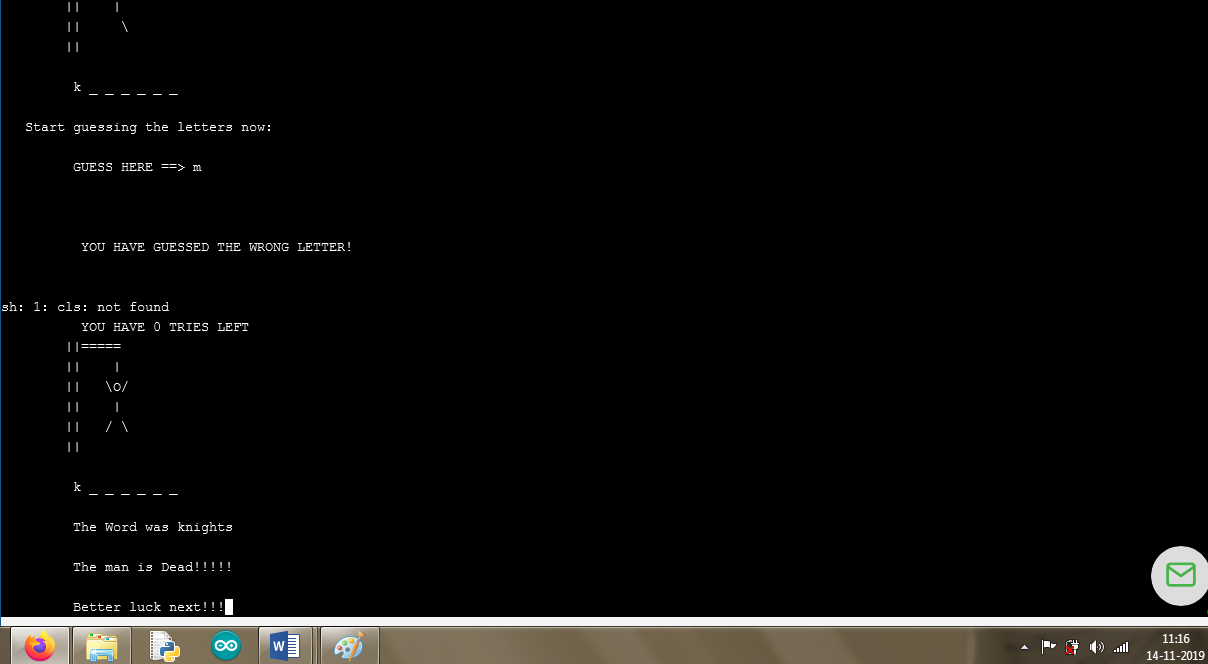


Fig 7.8 Output when all your guesses are over

**CHAPTER 8:**

**CONCLUSION**

Through all the research we have successfully created our desired game HANGMAN. This game uses single dimensional array and loops to get the input (the name of the movie) and process it to find out the number of letters in the name of the movie and display it to the second player.

In this project we learned how to utilize arrays and loops and manipulate then to our requirements. This way we explored various new ways and applications of arrays and loops which helped us to better understand the concept and expand the horizons of our knowledge. We learnt how to implement the switch case and use it to our advantage.

Our goal is achieved since we have coded a game which is suitable for all age groups as this helps them to be interactive through entirely of the game, and helps them to enhance their analytical skills and produce much quicker response to any questions whether it concerns gaming or real life experiments.

**DRAWBACK AND FURTHER IMPROVEMENT**

Although we have made a game considering all the possible outcomes of the games, there still are drawbacks that can be overcome in the future. Some of them are listed below.

1. The number of players are limited to two: The number of player are limited to two, although more the merrier.
2. Here in the game that we have created, it is limited to two players.
3. Considering it is a data structures project it only utilizes basic array data structures.
4. The origin of the game is not suitable for children
5. There is no visual appeal: this project does not use any GUI whatsoever.

**CHAPTER 9:**

**REFERENCES**

In this section we would like to mention the sources of the research and the information we collected. We would like to acknowledge all these sources that helped us during this project.

1. <https://www.geeksforgeeks.org/arrays-in-c-cpp/>
2. <https://www.codeproject.com/Articles/447332/Game-Programming-in-C-For-Beginners>
3. <https://www.geeksforgeeks.org/menu-driven-program-using-switch-case-c/>
4. <https://www.adweek.com/digital/social-game-features/>
5. <http://gamecodeschool.com/blog/making-games-where-do-i-start/>
6. <https://www.codingame.com/start>