**CHAPTER 1**

**INTRODUCTION**

* 1. **COURSE OBJECTIVES**

This project is built considering all the students and children around the world.

The main objective of this project are:

* To enhance their problem solving skills
* To top up their brain with IQs & non-bookish knowledge
* To increase their thinking power
* To make proper utilization of the leisure time
  1. **PROBLEM STATEMENT**

Games in this era are like more focused towards only fun and seeking attention. Students easily gets addicted towards such unreliable and non-beneficial games that does not have any purpose. Lacking general IQ, knowledge and less exposure to outside world is commonly seen in today’s peers. They being narrow minded, idle and socially isolated. These games consumes leisure time without leaving any knowledge and basic skills on the players which creates high impact on their career and personal development.

* 1. **OUTCOMES OF THE PROJECT WORK CARRIED**

The outcomes of the project work carried are:

* In-depth knowledge of all Object oriented Concepts.
* Able to solve the problems using OOPs concepts.
* Exposure to most of the syntax of C++.
* Enhanced with coding skills.
* Referred many online and offline materials.
* Proper utilization of Leisure time.
* Independent coding.

**CHAPTER 2**

**OBJECT ORIENTED FEATURES**

**2.1 CLASS**

Class in OOPs is a user-defined type. It is declared by using the keyword ‘class’. It has two compartments namely ‘Private’ and ‘Public’. It includes data members which are called class data member which comes under the private section of class and the functions which are called as class member functions that comes under the public section of the class. The data member in private section of class are protected and are not easily accessible by any other functions except the functions that are declared with in the public section of class. It is also called as the collection of objects. Its syntax is,

Class (classname){

Private:

(data members);

Public:

(member functions);};

**2.2 OBJECTS**

An object is instance of a class. As shown in Fig 2.1, we can create as my object as we want. Each object created will possess the values of all the data members of the corresponding class. We create an object and access the private members of the class with the help of object dot functions. All the member functions of a class is called using the object. Each objects shares the same member functions to access the data members of the class. A constructor is also called while creating an object. The syntax to create an object is;

<class\_name> <object\_name>;

Fig. 2.1: Class-object representation

**2.3 ENCAPSULATION**

Data encapsulation means the wrapping up of class data member and the class member function together into a single unit is known as encapsulation. Encapsulation means that we want to hide unnecessary details from the user. The Private data can be accessed only through those functions which are present inside the class but the functions declared outside the class can’t access the data members.

**HUMAN BEING**

nose, mouth, hands, ears, eyes, legs

walk, eat, sleep, watch, play

Fig. 2.2: Concept of Encapsulation

Here Fig 2.2 gives an example of encapsulation in which nose, mouth, hands, ears, eyes, legs, etc. are the properties or the data members and walk, eat, sleep, watch, play, etc. are the functions of HUMAN BEING.

**2.4 INHERITANCE**

Inheritance is the process in which object of child class will acquire/inherit the properties of parent class. Through this each derived class will shares the properties of base class. For example, let’s say Bird is a class and has the properties of feathers, wings and functions like laying eggs. Flying and Non- flying birds are also classes, but most of the properties and functions are included in parent class Bird. Hence, they can inherit everything from class Bird using the concept of inheritance. Fig 2.3 shows and example of hierarchical inheritance.

There are:

* Single Inheritance.
* Multilevel Inheritance.
* Multiple Inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance.
* Multipath Inheritance.

SOCCER PLAYER

pass

dribble

MIDFIELDER

pass

dribble

FORWARD

pass

dribble

DEFENDER

pass

dribble

Fig. 2.3: Hierarchical Inheritance

**2.5 ABSTRACTION**

Abstraction is the process of hiding irrelevant details from the user. This represents essential features without including the background details or information. For example; when we execute a program, we write the code after that we compile and press run, the console shows the output of the program, what actually happens when we compile and press run is hidden as it is not relevant to us.

**2.6 POLYMORPHISM**

Polymorphism simply means taking more than one form with the same name as shown in Fig. 2.4. In C++, a function with same name but different arguments can be termed as polymorphism. For example; consider the operation of addition, for the two numbers the operation will generate a sum but if the operands are string instead of numbers then the operation will produce a concatenated string.

There are usually two types of polymorphism:

* + 1. **Operator Overloading**

Operators can be overloaded using either friend function or simply member function. If the operator is unary then we must pass only one argument with friend function and should not pass any argument with member function. Similarly for binary operator we must pass only one argument with member function and should pass two argument through friend function.

* + 1. **Operator Overriding**

Operator overriding is a late binding which is performed at run time. More than two functions having same function name with same parameters is called operator overriding.

GAMES

Play ()

Object 1

Play (football)

Object 2

Play (hockey)

Object 3

Play (basketball)

Fig. 2.4: Concept of Polymorphism

**CHAPTER 3**

**PROJECT DESIGN**

**3.1 HARDWARE SPECIFICATIONS**

The hardware specifications are:

* Processor: Intel Pentium 4 or more
* Ram: 1 GB or more.
* Hard disk: 40 GB hard disk recommendation for the primary partition.

**3.2 SOFTWARE SPECIFICATIONS**

The software specifications are:

* Compiler: Turbo C++
* Operating System: windows 7 or later

**3.3 ALGORITHM**

Step 1: START

Step 2: Create a class and declare name[20] , wrong\_guesses and the member functions namely getdata(),display(),play(), winner(Dipesh, Dipesh) as friend function also overloaded functions secret\_word(char, char(), char()) and secret\_word(char(), char(), int).

Step 3: Initialize some constants max\_tries=5, max\_len=50.

Step 4: Create a two dimensional array words [][max\_len] in play() function to store collection of words.

Step 5: Read the name of the player and initialize wrong\_guesses to 0.

Step 6: Read the value of choice and using rand() function store the random word from the array into a variable word.

Step 7: Take the length of the word, encrypt it with ‘\*’ by the same length and assign it to variable encrypted\_word.

Step 8: Begin reading the guess from the player until gusses are used up.

Step 9: If the guess is correct fill the encrypted\_word with the same guess letter in the same position otherwise increment the no. of worng\_guesses.

Step 10: Check if guess matches by comparing word and encrypted\_word,

If(strcmp(word,encrypted\_word)==0){}

Step 10.1: If yes then display the message “you win’ and goto step 13

Step 10.2: If no then goto step 11

Step 11: If wrong\_guesses<max\_tries

Step 11.1: Then, goto Step 8

Step 11.2: Else goto Step 12

Step 12: Display the message “you lose”.

Step 13: END

**3.4 CLASS DIAGRAM**

Dipesh

(Base Class)

(data members);

char name[20];

int wrong\_guesses;

Polymorphism

friend function

Shrestha

(Derived Class)

single inheritance

Pure virtual function

Fig. 3.1: Class diagram

**3.5 DATA FLOW DIAGRAM**

Cin>>name

Int const max\_tries=5

Int const max\_len=50

Cin>>choice

initialize two dimenstional array

wrong\_guesses=0

check=0

i=0 i++

i<check\_length

n=rand()%choice

check\_length=strlen(words[n])

while(wrong\_guesses <max+tries)

word[i]=words[n][i]

encrypted\_word[i]=”\*”

cin>>guess\_letter

check++

i=0 i++

i<check\_length

if (word[i]==encrypted\_word[i])?

if (guess\_letter==word[i])?

encrypted\_letter[i]=guess\_letter

flag=0

if (flag!=0)?

wrong\_guesses++

cout<<word

cout<<”You Lose!”

if (check==0)?

cout<<”You Won!”

i=0 i++

i<check\_length

Fig. 3.2: Data Flow Diagram

**CHAPTER 4**

**IMPLEMENTATION**

Various OOPs concepts have been implemented into this project due to which it have been possible. Brain Teaser is normally a simple game which displays an encrypted word and the user has to think out of the box to decrypt it or simply guess it correct. A class with data members and member functions is create. This class keeps track of name of the player and the number of wrong guesses made by the player. There are two member functions with same function name but different arguments which defines the concept of polymorphism introduced into this project. One of the function is used to encrypt to be guessed word and the other function is used to make all kinds of comparisons between the guessed word by the user and the encrypted word. This class also includes friend function which compares between two players and filters out the winner who has less number of wrong guesses. The objects of the two players are passed into this function from which the number of wrong guesses made by the user is compared and winner is selected.

The Source code are as follows:

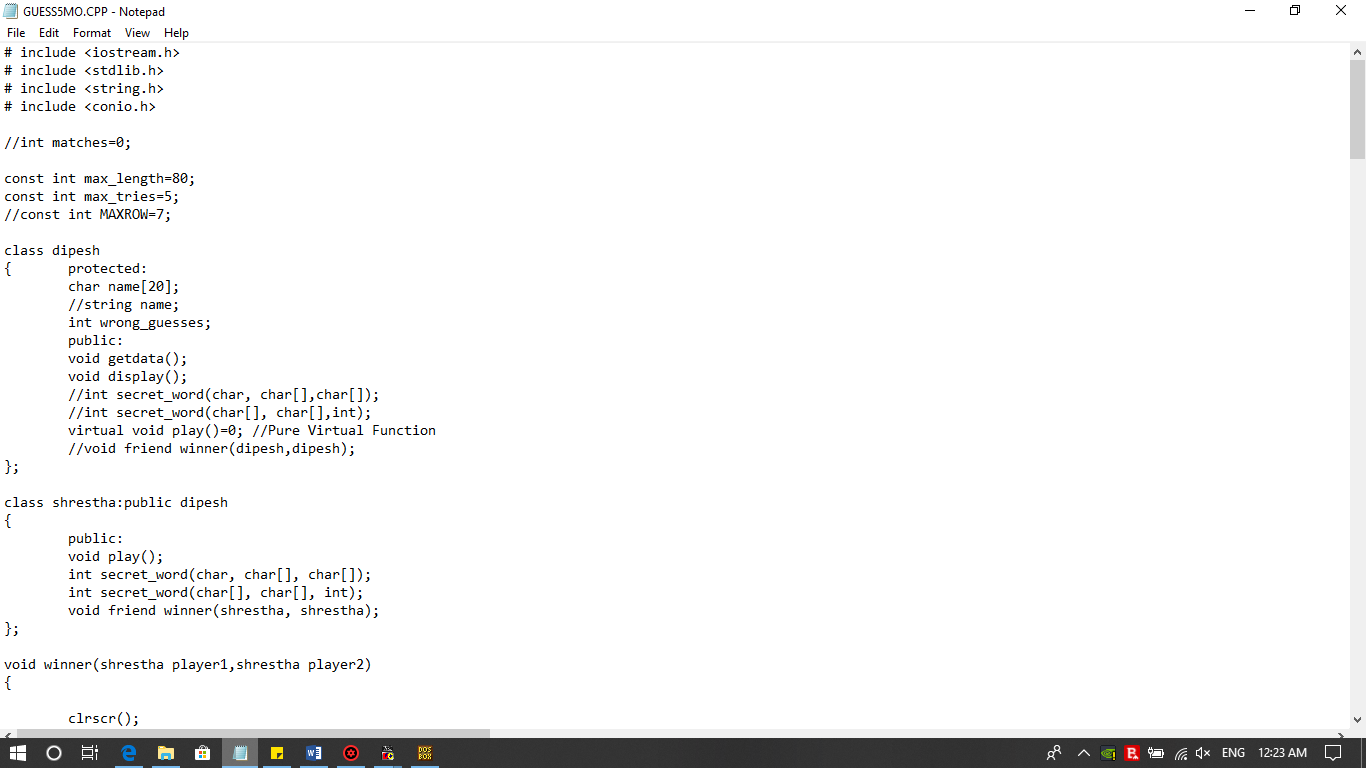


Fig. 4.1: Base class and pure virtual function declaration

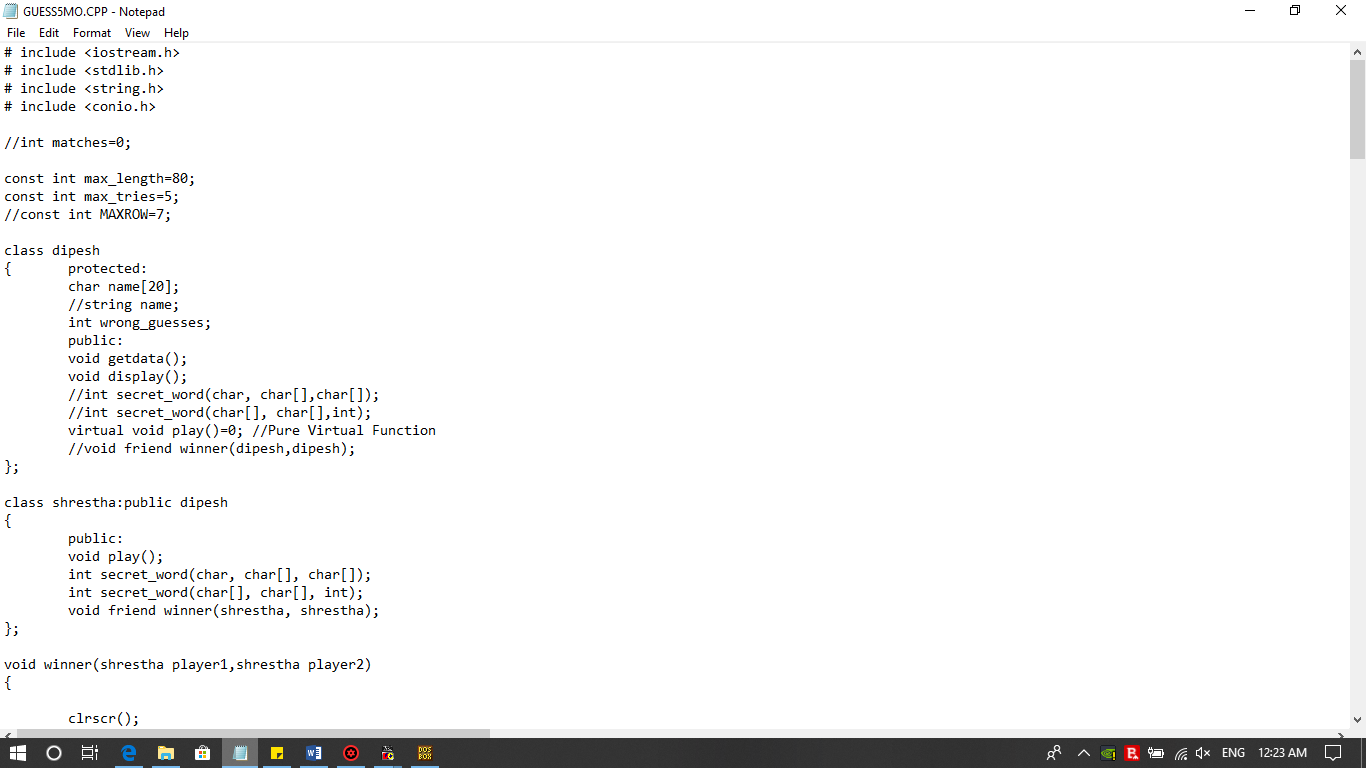


Fig. 4.2: Derived class and late binding conceptFig. 4.3: Friend function and early binding

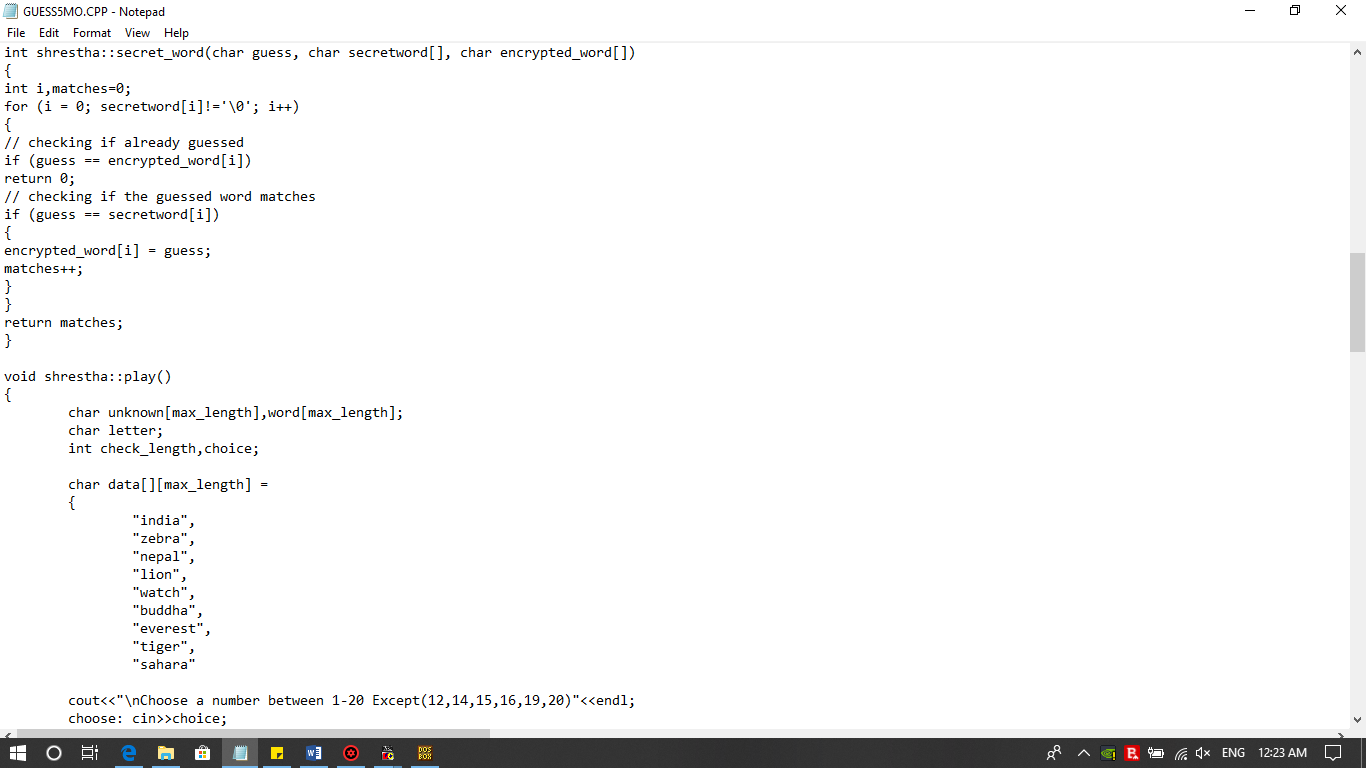
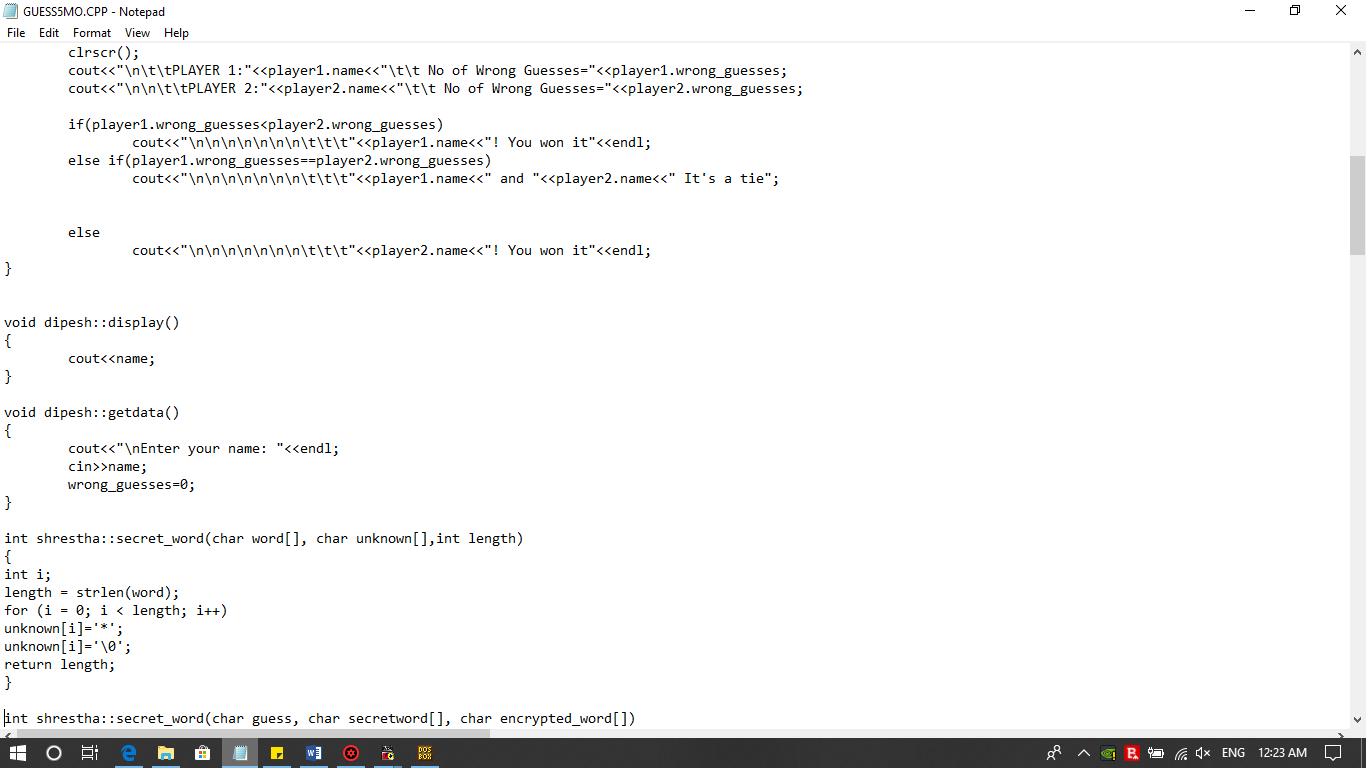


Fig. 4.4: Polymorphism and Pure virtual function definitionFig. 4.5: Rand() function and Switch-Case statements

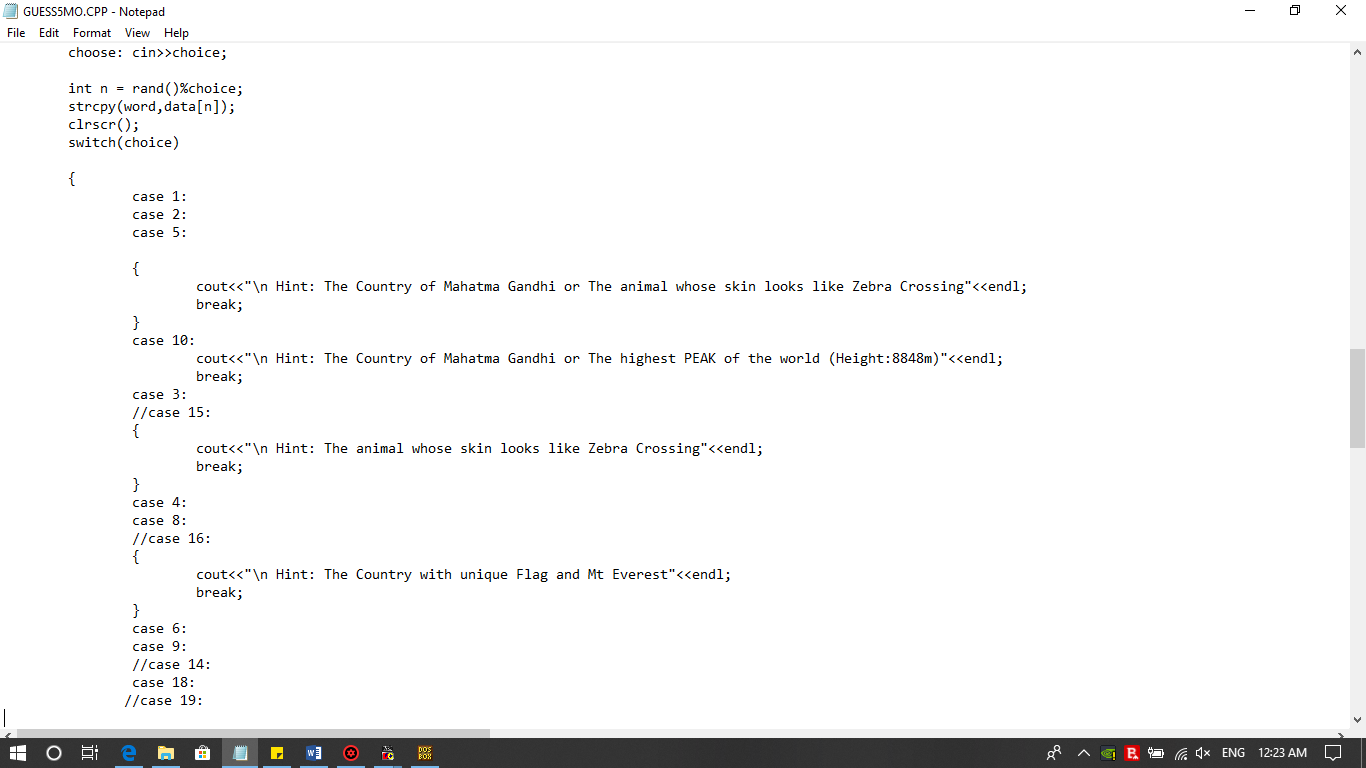
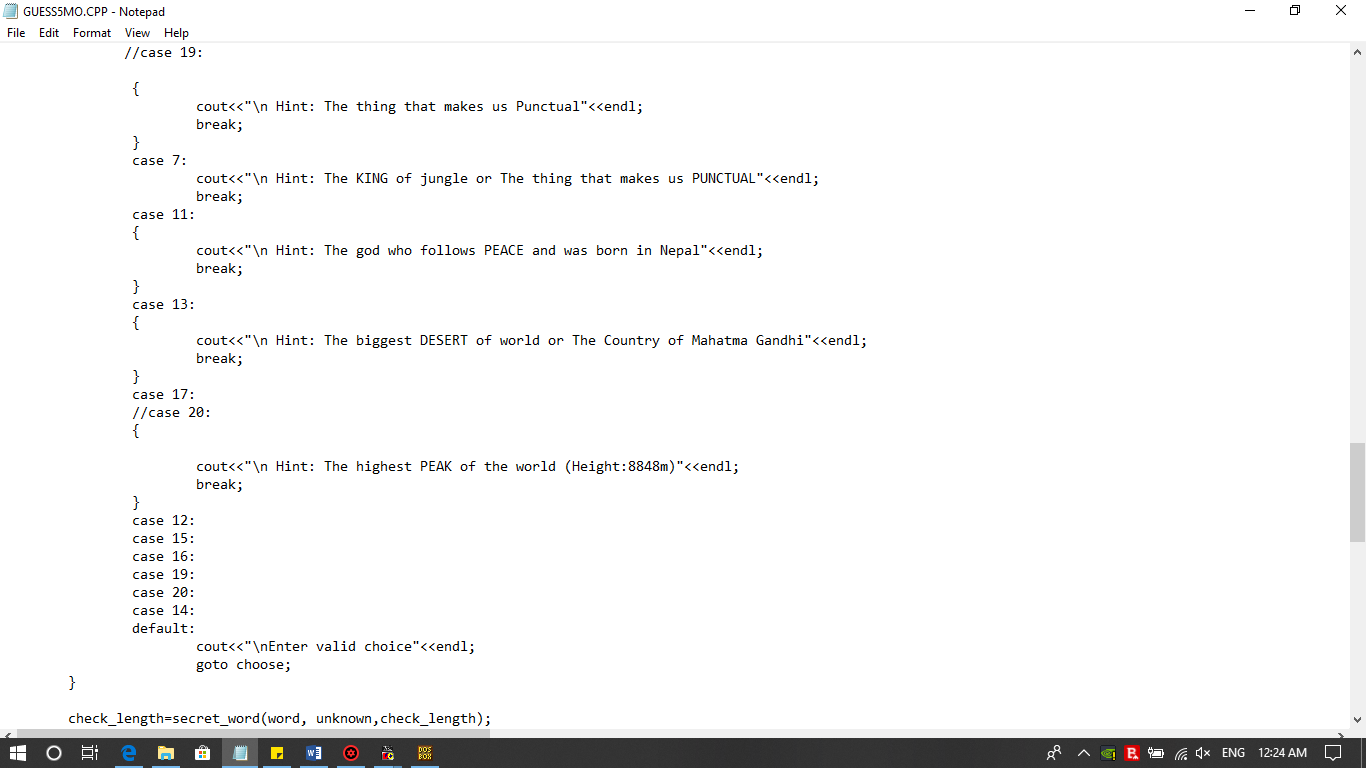


Fig. 4.6: Switch-Case statements



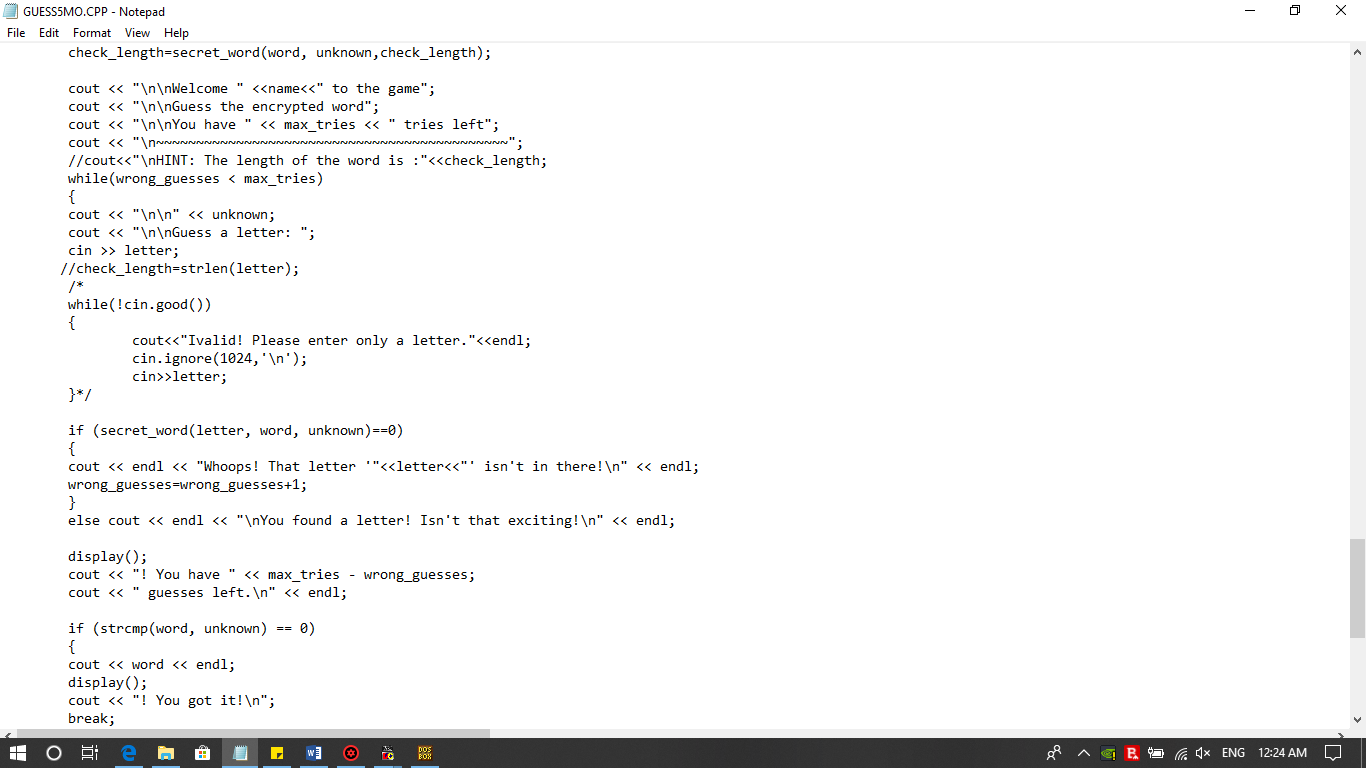
****

Fig. 4.7: Checking for match and wrong guesses

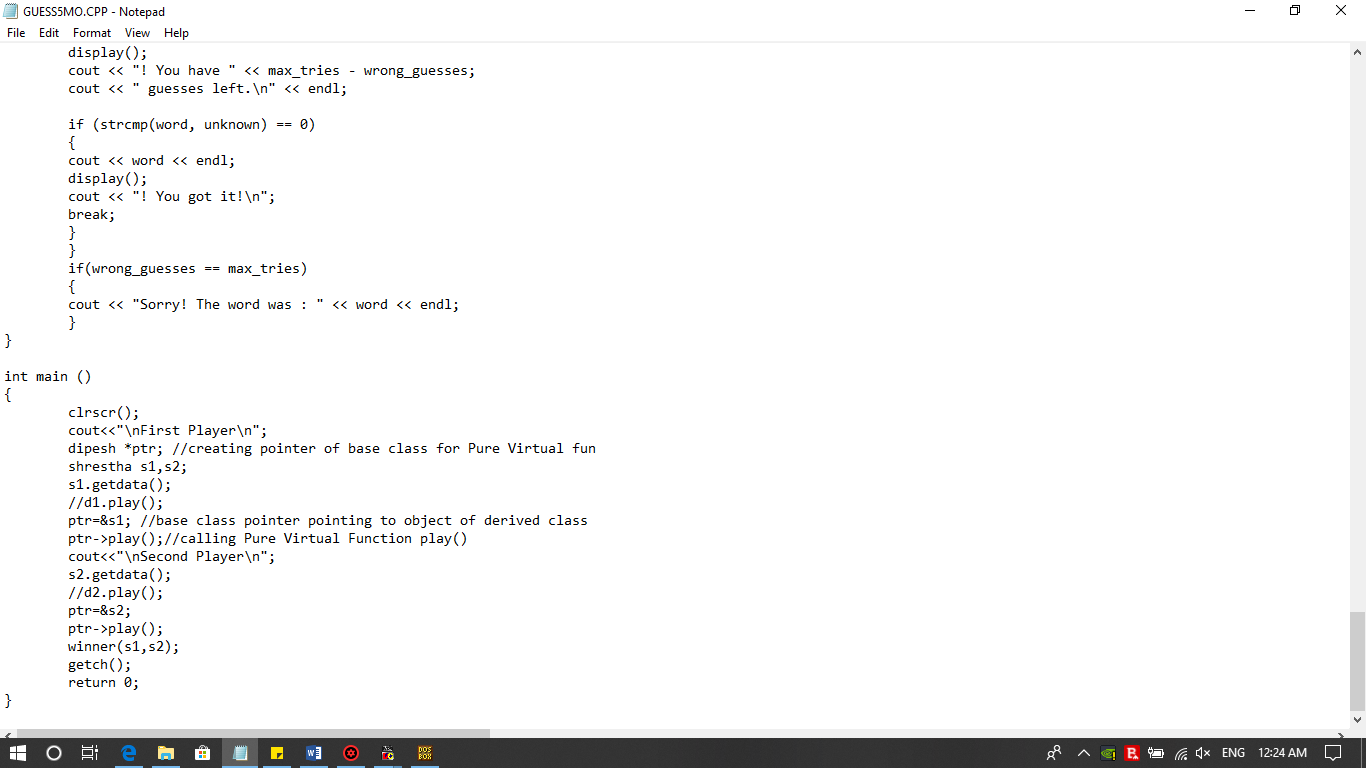


Fig. 4.8: Main program

**CHAPTER 5**

**OUTPUT SNAPSHOTS**



Fig. 5.1: Entry of First Player

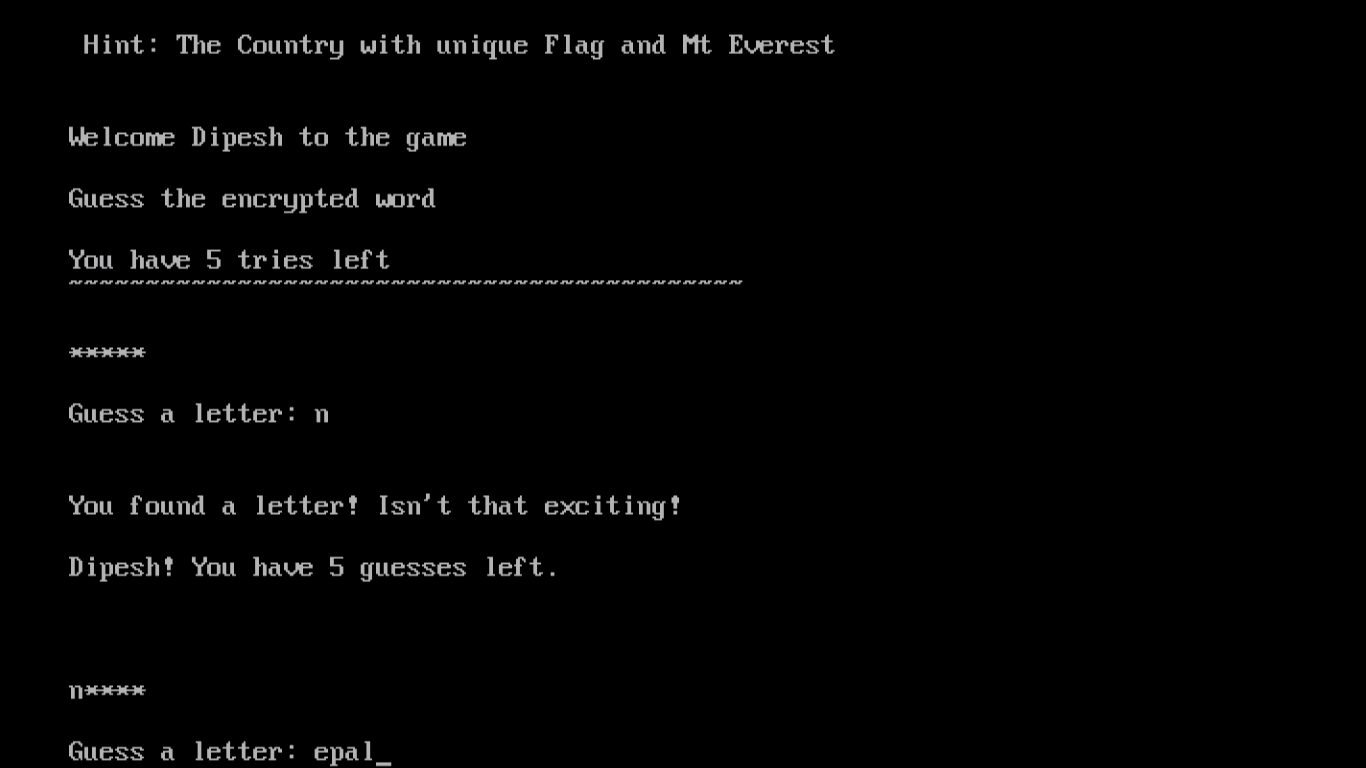


Fig. 5.2: Guess of First Player

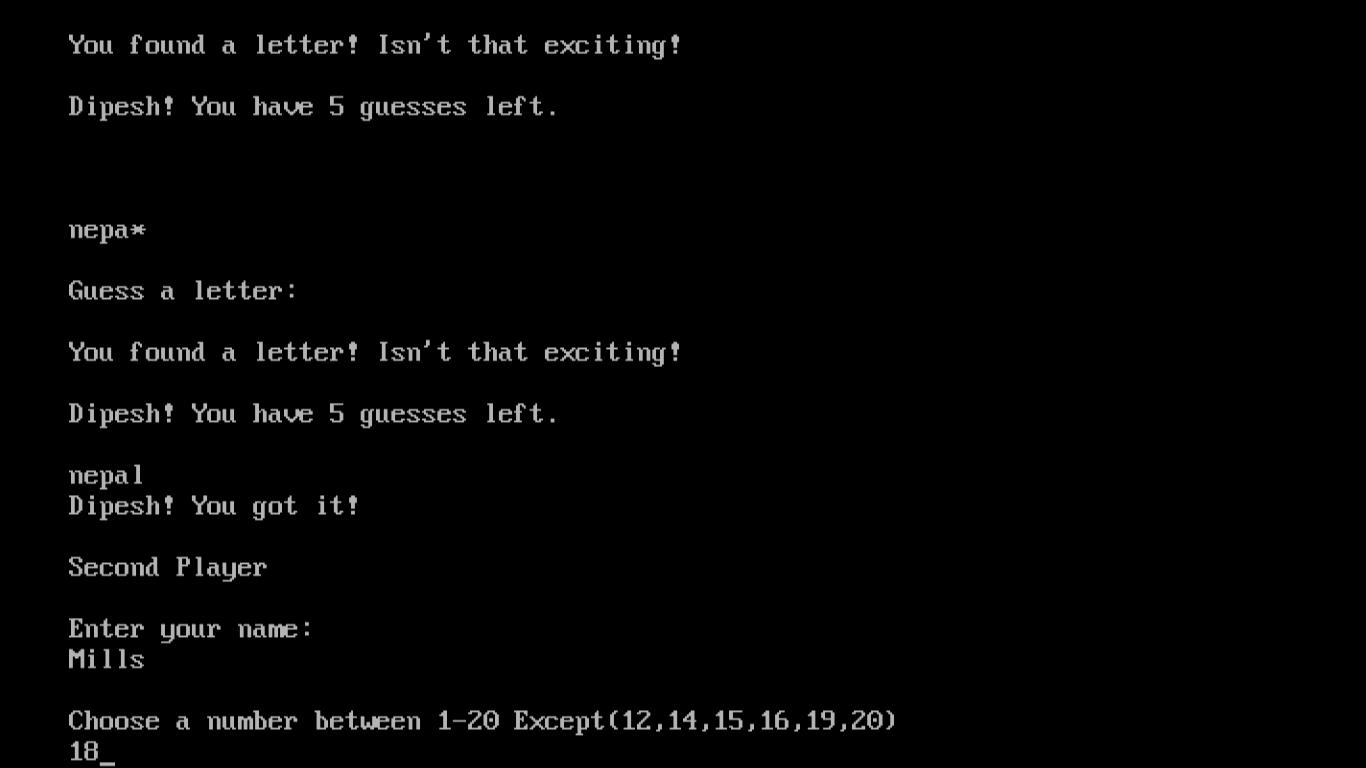


Fig. 5.3: Entry of Second Player

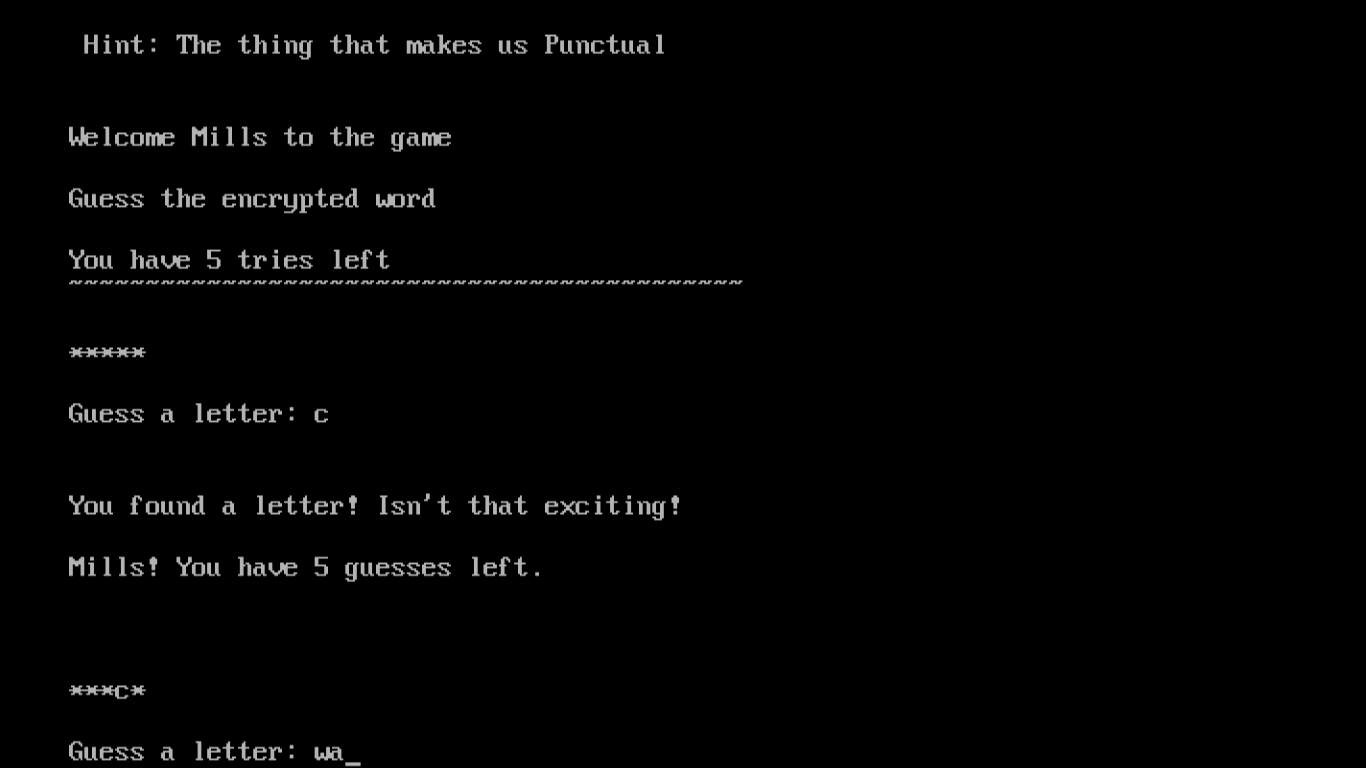


Fig. 5.4: Guess of Second Player

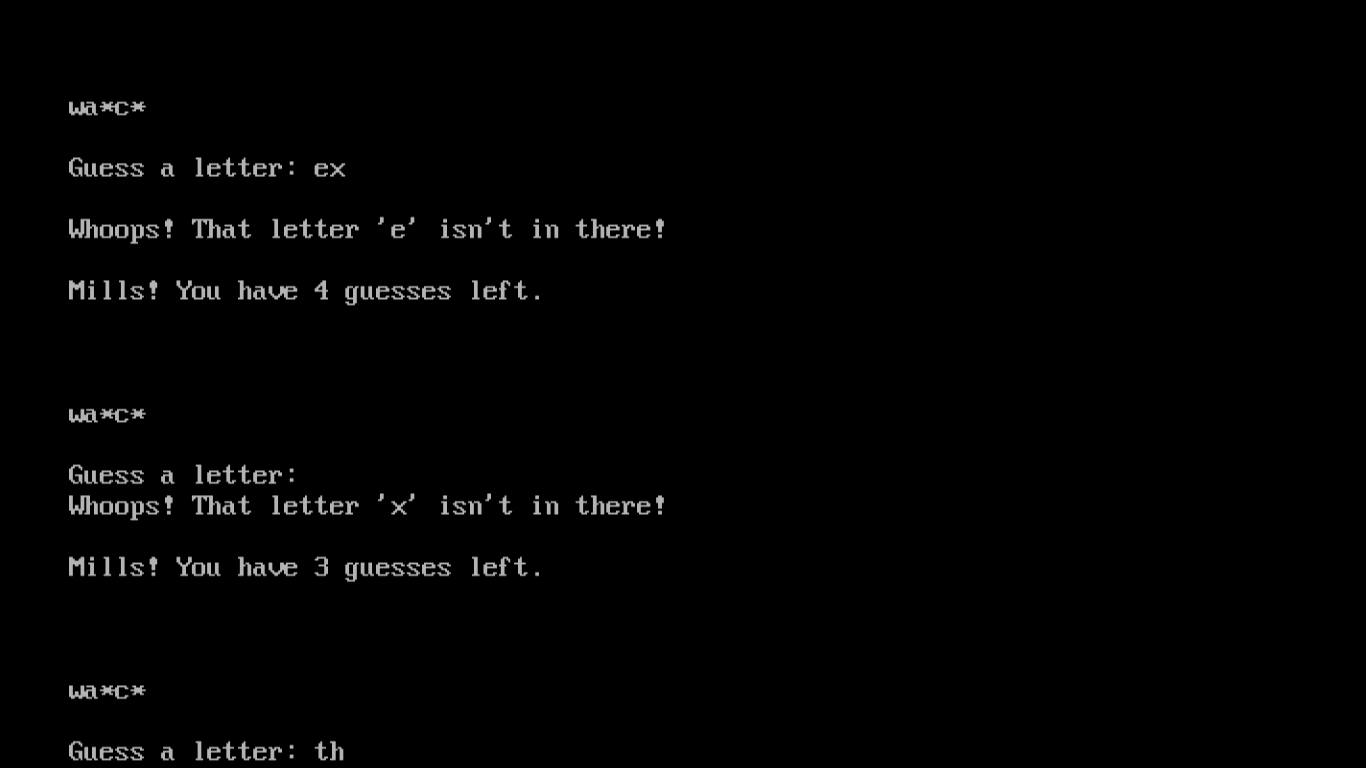


Fig. 5.5: Wrong guess of Second Player



Fig. 5.6: Result

**CHAPTER 6**

**CONCLUSION**

The main objective for creating this project was to create an awareness in students and coming generation’s children’s as they are more fascinated/ attracted towards game. There are many games which focuses only towards making the gamer enjoy their product. We can rarely see the games that focuses on increasing the learning skills, IQ knowledge or education related games. Playing games is not a bad thing, but playing games along with refreshing mind is always a better idea. Nowadays we can see a mobile phone in every students hand instead of books and copies. Games like pub g, mini military, fifa, fortnite, etc are teaching students the way to pass their time and take enjoyment. Minimizing more use of such games and seeking for alternatives is best for every student’s future.

Brain Teaser is also a game but has more learning skills, productivity, moreover focuses on IQ training of a students. We need more logical and problem solver for our future which is only possible when the students starts thinking for their problems on their own without their dependency on anyone. Exposure to such games from their childhood would make them creative from their young age. Simple IQ and general multiple choice questions must be much known to the children’s from their childhood. And this is determined by how they are spending their leisure time, how perfectly are students utilizing their leisure time. Brain Teaser requires thinking out of the box to make the answer correct, for this student will use their mind to think for the answer which will certainly help in the development of their mind as well as increase their remembering capacity. Thinking for the word which are encrypted is a tough task. Once students overcomes this task then they can be able to perform any logical tasks.

**REFERENCES**

* 1. <https://www.tutorialspoint.com/cplusplus/cpp_polymorphism.htm>
  2. <https://www.geeksforgeeks.org/friend-class-function-cpp/>
  3. Stephen Prata, “C++ Primer Plus”, Pearson Education Limited, 6th Edition, 2015
  4. Barbara Johnston, “C++ PROGRAMMING Today”, Pearson Education, 2nd Edition, 2015
  5. E Balagurusamy, “Object Oriented Programming with C++”, TMH, 6th Edition, 2013