PROJECT REPORT

Programming Fundamental

S

Hangman

Group Members

Areeba Azam  
Hasnain Ali  
Pir Ubaidullah Jan Sarhandi

**Problem:**

Guessing algorithms have long been around the IT industries. They were initially created for security and password protection purposes. Guessing algorithms can be used to create strong security for the protection of something, it can also be used for puzzling simulations that can also be for security purposes or for simple reasons such as entertainment. In entertainment purposes this algorithm can be used to create a variety of games and simulations to challenge the mind and have users thinking. This algorithm would follow the same concept of puzzles and have the users guessing and trying to solve the puzzles by trying a variety of different approaches towards the puzzle, until they get it right. Now for us the real problem was actually learning and understanding the logic of these puzzle/ guessing games and actually implementing it in our own program and in our own understood ways. We wanted to create a program that would have the user guessing the word, letter by letter until he gets it right or used up all his chances/life.

**Solution:**

The solution to the problem had to start from actually understanding the problem and have a broad approach towards figuring out its solution. Since there are multiple ways to solve a problem, we wanted to solve it using a way that was understood easily by everyone. After understanding the problem, we had to create our own logic for solving the problem so we created different scenarios and worked on those scenarios, step-by-step. We figured that first we needed to create a library of words to be guessed by the users, after that we wanted to create a randomizer to randomly choose one of the words from the library, we also had to randomize the sequence of the randomizer separately.

The basic solution was that the entered letters were compared to the randomly chosen word and each time a letter existed in the word it would show up in its respected place. On completion of the word, the user is prompted with a suitable response and then the game moves towards a different word. Multiple selection statements were used to handle the word and letter repetition and a switch case was used for the main menu of the game.

**Tools and Techniques**

**RNG (Random Number Generator):**

The first and most basic tool used in this program is the random number generators. The random number generator or ‘RNG’ for short is used for randomizing the words to guess from the specified library of words that are pre-built. It randomly chooses one word from the library each time the program is run, the sequence of these random words is also randomized. We also have made sure that during the entirety of 1 run/game, words aren’t repeated for example: if the word “MAN” has already been guessed once, it won’t be randomized another time during that run/game.

**Word Checking:**

This program/game is all about guessing the correct letters and finding out the correct word and this obviously required some logic to check if the words or letters being guessed by the user are even correct. Hence, we have used functions & multiple variables for checking and comparing purposes. As soon as the user inputs a guess letter, that letter is compared to the randomly guessed/chosen word and if the letter exists in the randomly guessed word, it shows up at its respected place. On correctly guessing all the letters of the word, the game moves onto the next word from the library.

For example: the chosen word was “MAN”

If the user entered ‘A’ it would show up as -A- and so on until the word is completed. On completion of the word “MAN” it will move to a different word.  
  
**Control Statements:**

Many control statements have been used in our program. Switch has been used to make the main menu of the program. Which has multiple choices, each leading to a different menu or function. Many if else statements have also been used for checking purposes and one if statement has been specifically used to check the entered letter from the user, for the specific purpose for quitting the game at any point during the gameplay. It’s made so that when the user types quit then the program ends.

**Graphics:**

We have included in our program in our own little artistic way by including different forms of graphics in it, such as: the loading screen of the game, the main menu of the game also includes graphics. The main graphics starts appearing when the user guesses wrong letters, on depleting all the lives/chances, a character is seen being hanged and that is the main graphics of the game and is also the reason this program/game is called “HANGMAN”.

**Architecture**

The program starts and a menu is seen on the screen with multiple choices.

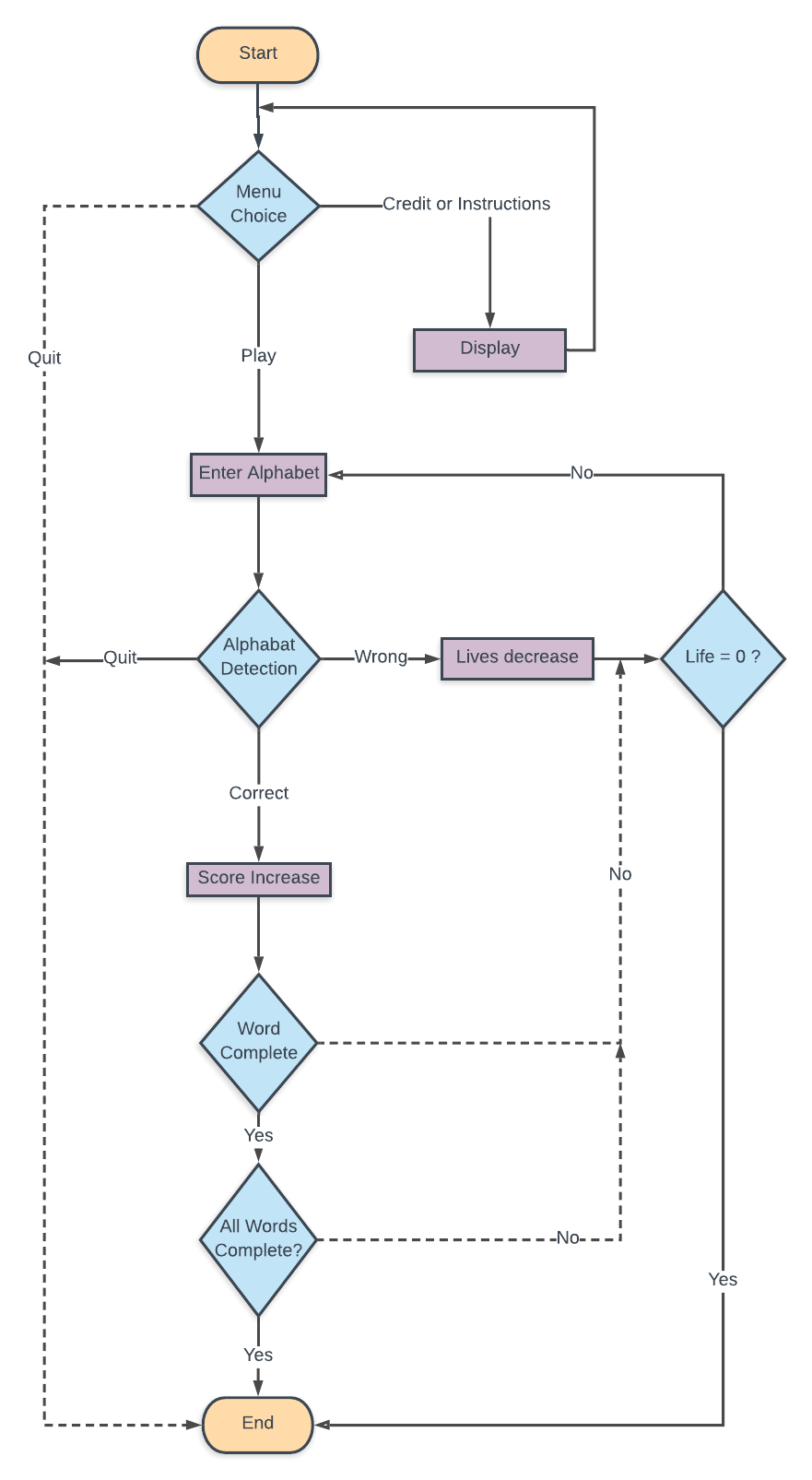
On entering the choices, the program moves towards that particular function of the game.

On entering the choice of credit or instructions, the user is shown the selected thing. The choice “Quit” is quite self-explanatory as it immediately quits the game and ends the program.

On selecting play, a word is randomly chosen and the user is shown a screen where he can start entering the guessed letter. Even at this point, if the user types “quit” the game is ended immediately.

On the guessed letter being incorrect, a life is decreased from the total lives i.e. 5 and if all the lives have been depleted the program moves towards the end.

Similarly, if the guessed letter is correct, the total score is increased. On guessing all the letters correctly, the program moves towards the next word and when all the words have been guessed the programmed ends.



**Result**

The result of all these different algorithms, logics, tools and techniques is a decent and entertaining puzzle game. Where the user can immerse himself in the game and challenge his mental abilities to guess the words whilst increasing his vocabulary.

The game starts with a menu with multiple choices and the game title being displayed and a character is also seen prompting the user to save him. The user can enter his/her choice at this point at the menu and move towards the selected feature. Credits, Instruction & quit are quite self-explanatory. Whereas talking about play, a random word is randomly picked from the pre-built word library/dictionary and the user is given the choice to enter her guess letters.

When the user guesses a right letter, the score is incremented and upon guessing the word correctly, the program picks out another random word and disregards this once so it can not be selected again. This cycle is repeated until all the words in the word library have been used up.

Upon entering a wrongly guessed letter the lives are decremented by one and some part of the hanged character graphics are printed/displayed on the screen. On depleting all the lives, the character is displayed hanged and dead. Prompting “You Died” and the program and all its loops come to an end.