PROJECT REPORT

Title: Hangman Logical Game

Introduction: The hangman logical game we have developed is a more user friendly version of Hangman. This game will not only make the user guess words (character by character), but it will also give the user certain hints, which will make guessing easier! Moreover, the game prevents repetition of mistakes, prints the wrong guesses made, and manages user high score. The aim behind this project was to basically use the x-86 architecture to develop a Hangman game, which is easy and feasible for the user to play.

Background: The research included testing basic Hangman programs which were already present in x-86, and looking for the missing functionalities as a user. Hence, we did some research on how these programs could be improved, what functions should be included in the game to make it a more interesting game, and we selected Hangman as our project accordingly.

Project Specification: The goal of this project is to make hangman a more interesting game for the user, and increase the difficulty level by the unpredictable length of words. This project will be faster and simpler than the existing systems for both the user, as well as the developer - as it has a minimal use of advanced functions. It will function just like a normal hangman game, with some advanced salient features.

Problem Analysis: While making this project, we had faced several problems, and complications. These problems were recognized by continuous testing of the program through various types of inputs, from the perspective of almost every type of user - and hence, they have been resolved accordingly. The major ones included:

- Storing, and updating high score in the file.
- Selecting a random word out of 40 words, and printing hints.
- Implementing Go-to-xy function in such a way that the Hangman picture, and the word to guess are placed on the same page, without any overlapping.

Solution Design: The project is made on x-86 architecture, with the use of basic built in functions present in the Irvine library. No additional libraries or assemblers have been used in completing the project, and therefore the project has been completed on Visual Studio.

The project comprises of a basic hangman game, with some exceptional functions which were not present in majority of the Hangman games made over Assembly language. Examples of these functions are High score management, increased interactivity with the user, and the print of an actual hangman printed alongside the user input to show the user the amount of chances left. Moreover, the program now has a wide range of words to be chosen through a random function. This decreases the probability of repetition of words. Also, the repetition of guesses has been controlled in this project – A user can not enter a same alphabet twice. If entered, the entry is not

counted as a valid entry, and error message is displayed. The array of wrong guessed characters is being printed on every wrong guess, to notify the user of the wrong guesses made, and promote experimentation learning through failed inputs.

To increase interactivity with the user, the pattern of the output screen has also been made attractive, which promotes the user to repeatedly play the game. High score management has also been added to act as a motivation for the user to play repeatedly, and set a new record!

Implementation and Testing: The implementation is done on Visual Studio, through the use of Irvine library only. The implementation has actually been done by following modular approach – that is breaking functions into simpler problems, and executing, and testing each part of the implementation, before executing the whole function. This was a core part of making the project. This took the greatest part of our time, as we had to implement and test the program back and forth before actually reaching to the desired output. We followed the modular approach to implement and test the program as it makes debugging easier, and it prevents complications in the code ahead.

Project breakdown structure: The workload was manageable, yet, quite lengthy. All members of the group have played an effective role in completing the decided task in the given time frame.

Day 1 – Mind mapping of functions to be made, and determining the flow of the program

Day 2, 3 and 4 – Samad initiated the Display functions, which included the front page, and the printing of Hangman on each guess.

Ashmal made the function to access words through random number generation through proper arithmetic, and the function to input words, and handle the string accordingly.

Hasnain looked for words, and made the function to print each word with their hints, and underscores instead of the words to guess.

Day 5 – Code debugging done by all of the members, where certain issues were highlighted. The issues included Go-to-xy errors where a certain entry would be over written, and another issue was that repeated entries were being allowed in the program.

Day 6, 7 – Samad worked on removing the repetitions, and making sure that no user had more than 6 chances.

Hasnain worked on the highscore management function, where a character was not being written in the file, so he had to come up with a logic to store the high score.

Ashmal worked on solving the display functions issues, and made sure that the program format was attractive for the user.

Day 7 – Testing and debugging done by all members to make sure that the program worked in an effective way.

Results: An interactive, user friendly game which provides the user an opportunity to think out of the box, as various types of words, with no common type have been given for the user to guess. This game is comparatively more user friendly, as it has a better interface, as well as the user can have a look at the instructions whenever desired.

Conclusion: As a conclusion, the hangman game we have made is having the same traditional way to operate, with additional features, which would encourage the user's interest in the game, and somehow promote learning of new words. The additional functions include better interactivity, adaptive learning and high score management. Repeated inputs are prevented, high scores are calculated based on the number of options left, and the length of character, and the user is continuously visualized with the hangman drawing on the right of the screen (as a reminder of the remaining chances).

Although, some additional features could have made the game more interactive such as adding a login panel, or adding the number of words that could be guessed. Moreover, the hangman visualization could have been made better. But these features could not be implemented due to our limited knowledge regarding the course, and the limitations in x-86 architecture. Summarizing, this project is one of the best user friendly versions of hangman.