VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

Mini project

On

"HANGMAN GAME"

By

SATHWIK	**4MT21IC043**
SHREESHA	**4MT21IC050**
MOHAMMAD HAFIZ	**4MT21IC028**
JNANESH	**4MT21IC018**
SOUNDARYA	**4MT21IC054**

DEPARTMENT OF COMPUTER SCIENCE (IOT) & ENGINEERING

(*Accredited by NBA*)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)
Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Badaga Mijar, Moodabidri-574225, Karnataka

2022-23

Abstract

The Hangman game is a classic word puzzle and guessing game that challenges players to guess a hidden word by suggesting individual letters. In this digital adaptation, players are presented with a blank word, the number of letters it contains, and a limited number of incorrect guesses allowed before a "hangman" figure is completed. Players must strategically guess letters to unveil the hidden word, using their linguistic and deductive skills. The game is engaging, educational, and can be enjoyed on various digital platforms, providing an enjoyable and mentally stimulating experience for players of all ages.

INTRODUCTION

. Get ready for a word-guessing challenge that will keep you on the edge of your seat. In this classic game, your task is to figure out the secret word one letter at a time. But be careful –each wrong guess, you'll draw a step closer to the hangman's noose. Can you decipher the word before it's too late? Type 'start' to begin the game and prove your word mastery! Good luck!

Purpose of the hangman Game System:

The Hangman game serves a multifaceted purpose that extends beyond mere entertainment. Primarily, it is a word puzzle game designed to challenge and engage the player's cognitive abilities, particularly their vocabulary and word-guessing skills. It encourages participants to think critically, make educated guesses, and strategize as they attempt to decipher the hidden word letter by letter. This intellectual stimulation is particularly valuable for educational purposes, as it aids in expanding one's vocabulary and enhancing linguistic capabilities.

Furthermore, Hangman fosters social interaction and communication. When played in a group setting, it becomes a collaborative activity that promotes teamwork and communication skills. Players take turns guessing letters and discussing potential word choices, fostering a sense of camaraderie and cooperation. It is often used as an educational tool in classrooms to reinforce language learning and to make the process of vocabulary acquisition more engaging.

On another level, Hangman also serves as a recreational pastime, offering individuals a fun and engaging way to pass the time. It can be played casually in various settings, from family gatherings to long car trips, providing a source of entertainment that requires minimal equipment and is accessible to people of all ages.

In a broader context, the Hangman game can be seen as a symbol of the power of words and the importance of effective communication. The goal of the game is to avoid the figurative "hangman's noose" by correctly guessing the word. This metaphorical representation underscores the significance of clear and precise communication in our daily lives, reminding us of the consequences of misunderstandings and miscommunication.

In essence, the Hangman game transcends its role as a simple word puzzle and emerges as a versatile tool for education, social interaction, and leisure, while also offering a subtle reminder of the importance of language and effective communication in our society.

Key Features:

Key features of a Hangman game typically include the following elements that make the game engaging and enjoyable:

- **Word Selection**: The game should have a mechanism to select a secret word from a predefinedlist or dictionary. The word to be guessed is usually chosen randomly.
- **Hidden Word Display**: The game displays the secret word as a series of dashes or underscores,with each dash representing a letter in the word. Initially, the entire word is hidden.
- **Letter Guessing**: Players can guess letters one at a time by inputting their guesses through a userinterface.
- **Incorrect Guess Tracking**: The game keeps track of the number of incorrect guesses made bythe player. This is often represented by drawing a hangman figure piece by piece as incorrect guesses accumulate.
- **Correct Guesses**: When a player guesses a letter correctly, the game reveals all instances of thatletter in the hidden word. Correct guesses help players uncover the word.
- **Incorrect Guess Penalty**: When a player guesses a letter incorrectly, the game penalizes them byadding a part to the hangman figure. Once the hangman is complete (usually after a certain number of incorrect guesses), the game is over, and the player loses.
- Winning Condition: The game checks for a winning condition, which occurs when the playersuccessfully guesses all the letters in the word before the hangman figure is fully drawn.
- Losing Condition: If the player runs out of attempts (i.e., the hangman figure is fully drawn), thegame ends, and the player loses.
- Feedback and Messages: The game provides feedback to the player after each guess, indicatingwhether the guess was correct or not. It also displays messages for winning or losing the game.
- Scorekeeping: Some Hangman games keep track of the player's score, which may includestatistics such as the number of wins and losses.
- Word Categories: In some versions of the game, words may be categorized (e.g., animals, fruits,countries), and players can choose a category before playing.
- Difficulty Levels: To cater to players of different skill levels, the game may

offer multipledifficulty levels, with easier or more challenging word choices.

- User Interface: The game provides a user-friendly interface that allows players to input letters,view the hidden word, and see the hangman's progress.
- Sound and Graphics: To enhance the gaming experience, Hangman games often include sound

effects, animations, or graphics to represent the hangman figure and other elements.

- Multiplayer Options: Some Hangman games offer multiplayer modes where players can taketurns guessing or compete against each other.

These key features combine to create an engaging and enjoyable Hangman game that challenges players' word-solving skills and provides entertainment.

Benefits of the Hangman Game System:

The Hangman game offers several benefits, making it a popular and enjoyable pastime. Here are some of the key benefits of playing Hangman:

- **Vocabulary Building**: Hangman exposes players to a wide range of words, helping them expandtheir vocabulary and improve their word recognition skills.
- **Spelling Practice**: The game reinforces correct spelling as players attempt to guess the letters inthe hidden word.
- **Critical Thinking**: Players must use deductive reasoning and critical thinking to make educatedguesses about the hidden word based on the letters they've guessed and the context of the word.
- **Problem Solving**: Hangman encourages problem-solving skills as players work to decipher thehidden word from limited information.
- **Patience and Perseverance**: Players learn patience and perseverance as they continue guessingeven after making incorrect guesses. It teaches that persistence can lead to success.
- **Educational Tool**: Hangman can be used as an educational tool in classrooms to teachvocabulary, spelling, and word associations in an interactive and engaging manner.
- **Stress Relief**: Playing Hangman can be a fun and stress-relieving activity, providing a mentalbreak from daily routines.
- **Language Learning**: For language learners, Hangman can be a valuable tool for practicing newwords and improving language skills.
- **Social Interaction**: In multiplayer settings, Hangman encourages social interaction andcommunication as players take turns guessing letters and words.
- **Entertainment**: Hangman is primarily an entertaining game that offers a mix of challenge andfun, making it a popular choice for leisure.
- **Cognitive Development**: It stimulates cognitive development by requiring players to use theirmemory, logical reasoning, and deduction skills.
- **Quick Gameplay**: Hangman games are relatively short and can be played in a short amount oftime, making them ideal for quick breaks or as a casual game.
- **Customization**: Many Hangman games allow players to customize settings, such as wordcategories or difficulty levels, to tailor the game to their preferences.

- **Competitive Spirit**: In multiplayer modes, Hangman can foster healthy competition amongfriends or family members, adding an extra layer of excitement.
- **A Sense of Achievement**: Successfully guessing a word in Hangman provides a sense of accomplishment and satisfaction, boosting self-esteem.

Overall, Hangman is a versatile and engaging game that offers a wide range of cognitive, educational, and entertainment benefits. Whether played individually or with others, it can be a rewarding and enjoyable experience.

IMPLEMENTATION

Code:

```
//HANGMAN GAME CODED USING C
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <time.h>

//structure to store a word of length 30 charcters.
typedef struct
{
  char word[30];
} words;

words wordlist[1000];          //array of stucture "words" used to store words upto 1000 words each of length 30.
char key[30];                  //array of characters used to store all the letters guessed by player.
int keys = 0, done;            //"keys" keeps track of number of guesses, "done" is a flag used to indicate completion of game, "done=1" means game is underway.

/*function to get a random index from the array of structure "words" containing words depending on the paramenter we pass into the function
 1 means we add all the words from the file and take custom inputs from player.
 2 means we add only words from the file to the array.
 3 we add only the words taken fromthe user and do not add words from the file.
  everything else is considered as exiting the game.*/
```

```
int
getRandomWord (int choice)
{
  int len, index, x = 0, n = 0;
   FILE *ptr = fopen ("words.txt", "r");

   switch (choice)
    {
    case 1:                    //it's left empty without break so that it executes case 2 and case 3.
    case 2:
     if (ptr == NULL)
{
  printf ("no such file."); // in case we do not find the file of words inform and exit the game.
  exit (0);
}
     char buf[100];                // words are copied into this buffer from file before adding into array.
     while (fscanf (ptr, "%s ", buf) == 1)
{
  strcpy (wordlist[n].word, buf); // adding all words from the file into the array of words.
  n++;
}
     if (choice == 2)              // we break the switch inside if condition because we may want it to execute case 3 also, if user passed case 1.
{
  break;
}
```

```
    case 3:
   printf ("Enter the number of words to add: ");
    scanf ("%d", &x);
      printf ("\nEnter %d words:\n", x);
      for (int i = n; i < n + x; i++)
{
  scanf ("%s", wordlist[i].word);
  len = strlen (wordlist[i].word);
  for (int j = 0; j < len; j++)        //loop used to verify that custom word contains only alphabets.
    {
      if (!isalpha (wordlist[i].word[j]))
{
  printf
    ("\n****ONLY ALPHABETS PLEASE RENTER THE WORD****\n");
     i--;
}
    }
}
      n += x;
      break;
    default:
      exit (0);
    }
  srand (time (NULL));              //seed random fuction with time so every new run gives new value.
  index = rand () % n;              //get random number and mod with "n" (number of words).
  return index;                     // return "index" which is now a random number between 0
```

```
and n.
}

/*Function to draw stickfigures given the word
  the guesses are stored in a public array named "key" and the length of the array is stored in a variable named "keys"
  these are compared with our random word kept track of by the paramter passed into the function "index"
  then the appropriate stickfigure is drawn depending on the number of "strikes"*/
void
stickFigure (int index)
{
  int strikes = 0, len = strlen (wordlist[index].word), count = 0, flag = 0;
  for (int i = 0; i < keys + 1; i++)
    {
      for (int j = 0; j < len; j++)
{                    //each guess is compared with each letter of the random word, both are converted to uppercase first.
if (toupper (key[i]) == toupper (wordlist[index].word[j]))
    {
      flag = 1;              //"flag=1" if guess matches letter from random word.
    }
}
      if (flag)
{
  count++;                   //"count" stores the total number of correct guesses.
  flag = 0;
}
```

```
    }
  strikes = keys + 1 - count;       //"strikes" stores the total number of wrong guesses so far.
  switch (strikes)
    {                               //each case contains the specific drawing for that many strikes.
    case 0:
      printf
("\n\n    +---+\n    |  |\n        |\n        |\n        |\n        |\n =========");
      break;
    case 1:
      printf
("\n    +---+\n    |  |\n    O   |\n        |\n        |\n        |\n =========");
      break;
    case 2:
      printf
("\n    +---+\n    |  |\n    O   |\n    |   |\n        |\n        |\n =========");
      break;
    case 3:
      printf
("\n    +---+\n    |  |\n    O   |\n   /|   |\n        |\n        |\n =========");
      break;
    case 4:
      printf
("\n    +---+\n    |  |\n    O   |\n   /|\\  |\n        |\n        |\n =========");
      break;
    case 5:
      printf
("\n    +---+\n    |  |\n    O   |\n   /|\\  |\n   /    |\n        |\n =========");
```

```
        break;
      case 6:                    //last case should display the random word and indicate to player that game is over.
        printf

("\n    +---+\n    |   |\n    O   |\n   /|\\  |\n   / \\  |\n        |\n =========\n");
        printf ("\nTHE WORD WAS : %s\n", wordlist[index].word);
        printf
("\n**************************************************************************
****\n");
        printf
("*********************************GAMEOVER*****************************
*****\n");
    printf
("**************************************************************************\
n");
        done = 0;
        break;
      default:                   //some how "strikes" value does not lie between 0 and 6 then execute this case.
        printf ("\n\n***ERROR STRIKES EXCCEDED***\n\n");
   }
    }

    /*Function to display the word given index
     we also need to display the word with blanks for undiscovered letters and letters for already guessed characters
     this function checks with array "key" and displays the word appropriately with blanks and letters.
     this function also checks for victory condition, if there are no blanks it means the player
```

```
has guessed all letters
  therefore we need to end the game and announce victory.*/
void
displayWord (int index)
{
  int len = strlen (wordlist[index].word), flag = 1;         //resets flag value to 1 every call.


printf ("\t");
  for (int i = 0; i < len; i++)
    {
      for (int j = 0; j < keys + 1; j++)
{
  if (toupper (key[j]) == toupper (wordlist[index].word[i]))
    {
     printf ("%c ", key[j]);           //displays correctly guessed letters.
      goto label;    //used to skip over setting of flag and printing a blank.
    }
}
      flag = 0;                          //"flag=0"indicates there are still letters to be guessed.
      printf ("_ ");            //prints a blank space for undiscovered letter.
   label:continue;
    }
  printf ("\n");
  if (flag)                     //checks if victory condition is fulfilled.
    {
      printf
("\n\n**********************************************************************
******\n");
```

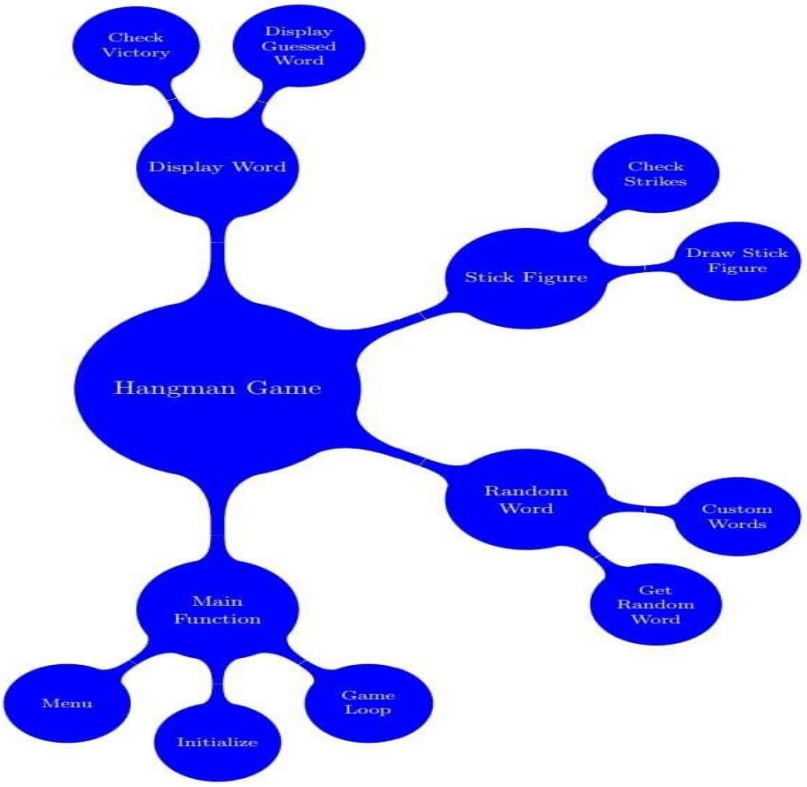
```
        printf                                  ("**********************************YOU
    WON!!**********************************\n");
        printf
    ("*****************************************************************************
    **\n");
        done = 0;                   //denotes end of game and player has won.


      }
    }


  void
main ()
    {
      int index, len, choice;
      char userkey;
      printf
    ("*****************************************************************************
    **\n");
      printf                                ("*****************************HANGMAN-
    GAME*********************************\n");
      printf
    ("*****************************************************************************
    **\n");
      printf ("**                  _____________________                  **\n");
      printf ("**                  |         |         |                  **\n");
      printf ("**                  |         |                            **\n");
      printf ("**                  |        _|_                           **\n");
      printf ("**                  |      .-"""""""-.                      **\n");
      printf ("**                  |    .'         '.                      **\n");
```

```
 printf ("**                  |   [   O      O   ]                    **\n");
printf ("**                   |   :              :                    **\n");
printf ("**                   |   |              |                    **\n");
printf ("**                   |   : ',         ,' :                   **\n");
printf ("**                   |    {  '-......-'  }                   **\n");
printf ("**                   |     '.         .'                     **\n");
printf ("**                   |       '-......-'                      **\n");
printf ("**                   |                                       **\n");
printf ("**                 __|_____________________________|                **\n");
printf ("**                 ---------------------------                    **\n");
printf ("**                                                           **\n");
while (1)
 {
   printf ("**                      *****MENU*****                       **\n");
   printf ("**                 ____________________________________________          **\n");
   printf ("**                |                                    |          **\n");
   printf ("**                | PRESS 1 to ADD custom words.       |          **\n");
   printf ("**                | PRESS 2 to PLAY.                   |          **\n");
   printf ("**                | PRESS 3 to play ONLY with custom words. |          **\n");
   printf ("**                | PRESS ANYKEY to EXIT.              |          **\n");
   printf ("**                |____________________________________|          **\n");
   printf ("**                                                           **\n");
   printf ("**                                                           **\n");
   printf ("**                ENTER YOUR CHOICE : ");
   scanf ("%d", &choice);printf ("       **");
   printf ("**                                                           **\n");
   printf ("**                                                           **\n");
```

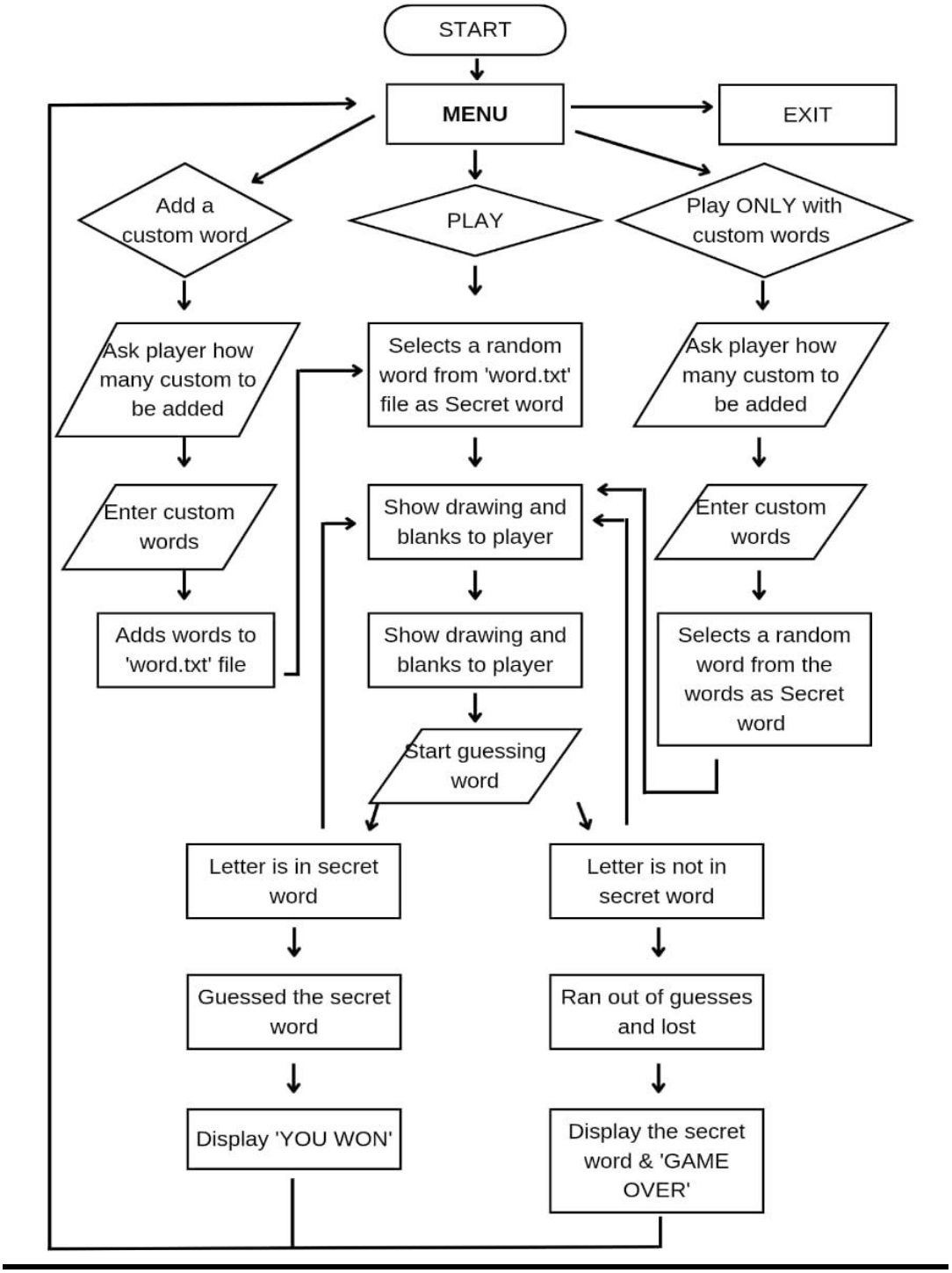
```
printf("**************************************************************************
*******\n");
  printf
("**************************************************************************
**\n");


    index = getRandomWord (choice);
    len = strlen (wordlist[index].word); //store length of random word.
    printf ("\nSTART:\n");
    printf
("\n\n   +---+\n   |  |\n      |\n      |\n      |\n      |\n =========");
    printf ("\t");
    for (int i = 0; i < len; i++)
{
  printf ("_ ");
}                    //until here we setup the initial stage for the player to input the first guess.
    done = 1;
    while (done)          //we run while loop until game ends by player guessing right or
running out of tries.
{
printf ("\n\nGUESS: ");
  scanf (" %c", &key[keys]);     //takes player guess.
  stickFigure (index);     //draws stickfigure and checks for loss.
  if (done)
    {
      displayWord (index);          //displays word and checks for victory
    }
  keys++;
```

```
}
    keys = 0;                    //resets guesses for next game.
  }
}
```

MIND MAPPING



FLOW CHART


START
MENU
EXIT
Add a custom word
PLAY
Play ONLY with custom words
Ask player how many custom to be added
Selects a random word from 'word.txt' file as Secret word
Ask player how many custom to be added
Enter custom words
Show drawing and blanks to player
Enter custom words
Adds words to 'word.txt' file
Show drawing and blanks to player
Selects a random word from the words as Secret word
Start guessing word
Letter is in secret word
Letter is not in secret word
Guessed the secret word
Ran out of guesses and lost
Display 'YOU WON'
Display the secret word & 'GAME OVER'

FUNCTION FLOW

Main Function (int main()): Displays the welcome message and menu options.

- Waits for user input to choose an option.
- Based on the user's choice, it can:
- Start the game (press 2).
- Add custom word (press 1).
- Play only with customized words (press 3).
- Exit the game (press any key to exit).
- Selects the random word from the “words.txt” file or custom words (depending upon the choice of the user.) as a secret word.
- Displays drawings and blank spaces .
- Player starts guessing letters for the secret word.
- If the player guesses the letter which is present in the secret word, that letter is displayed in its respective blank space.
- If the player guesses the wrong letter ,player loses one chance and is displayed in the drawing.
- If the player guesses all the letters of the word then he has won the game.

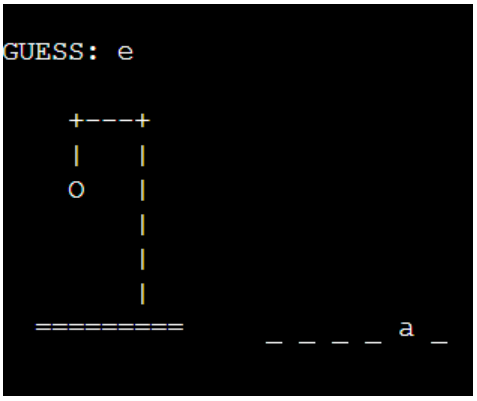
RESULT

Main Menu:

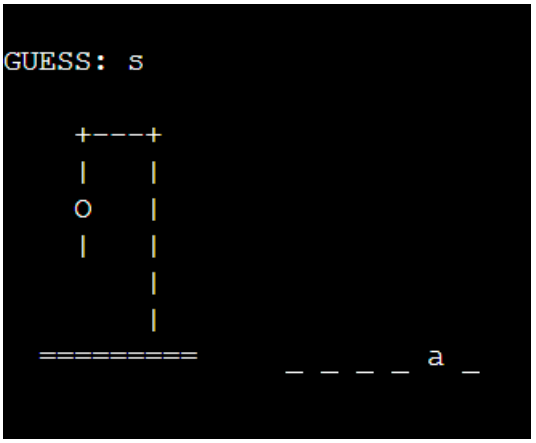
When we select PLAY:

Guessing a letter:

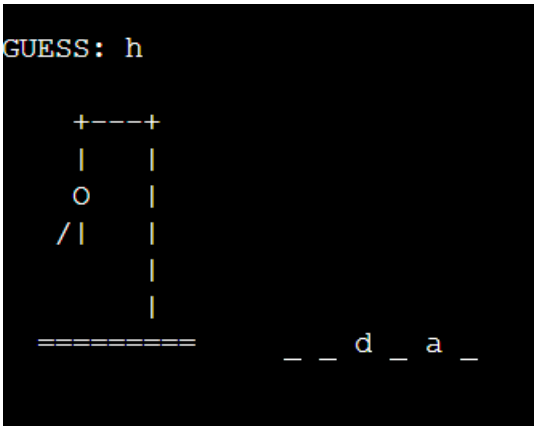
Wrong guess 1:



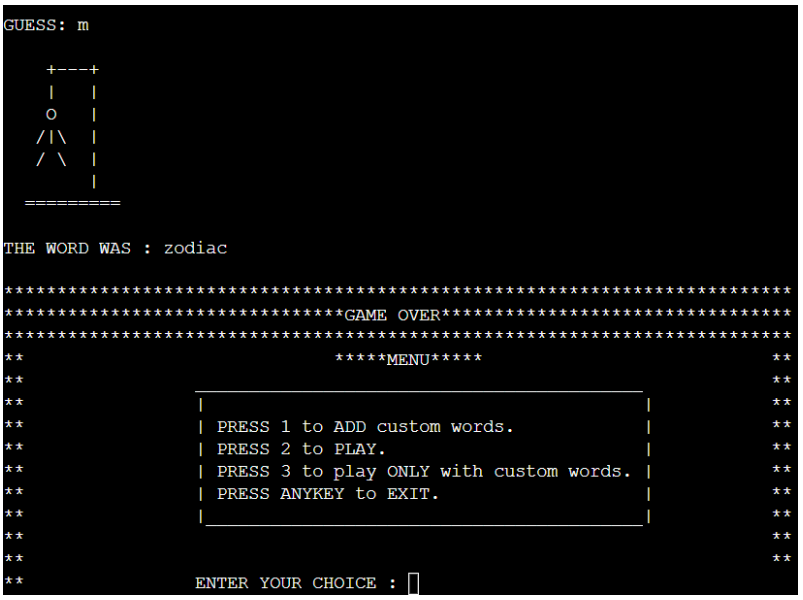
Wrong guess 2:



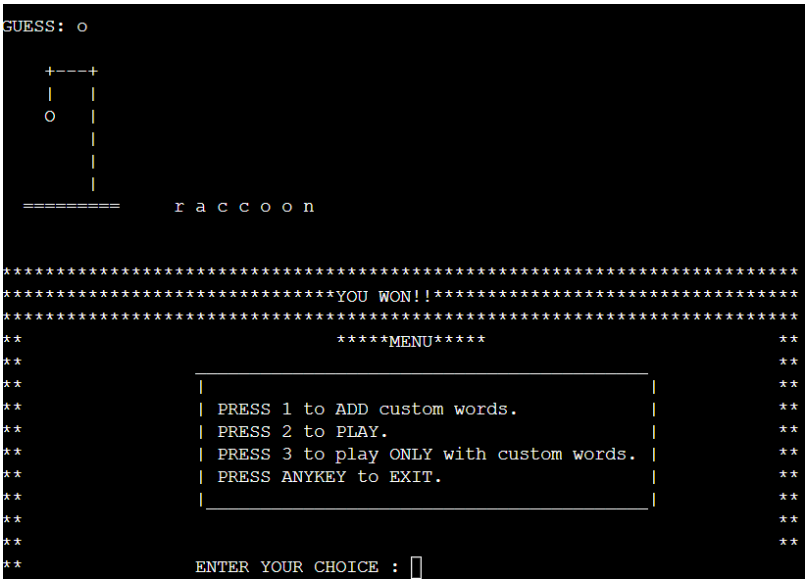
Wrong guess 3:



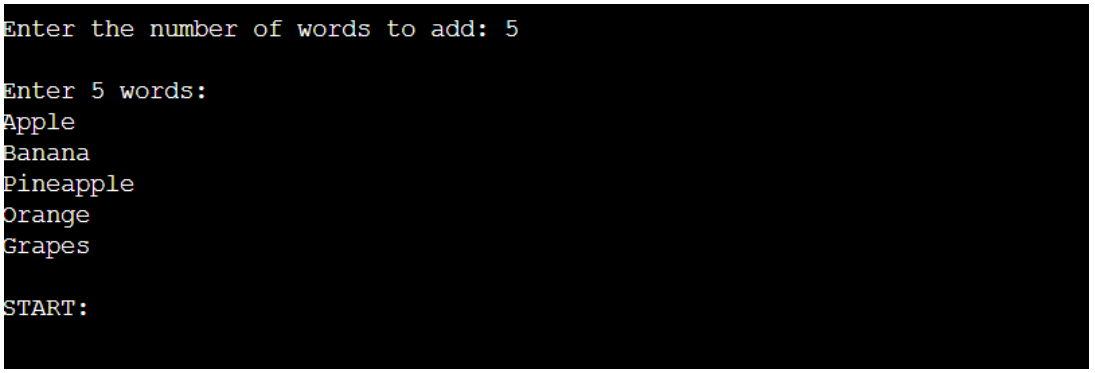
Out of chances:



Correct guess:



Adding custom words:



FUTURE ENHANCEMENT

Future Enhancements:

- When considering future enhancements for your Hangman code report, you can focus on several areas to provide a more comprehensive and informative document. Here are some ideas for future enhancements:
- **Code Optimization**: Discuss potential ways to optimize the code for efficiency and readability. Explore advanced C programming techniques, such as using data structures like linked lists for the word list or implementing more efficient algorithms for word selection.
- **User Interface**: Enhance the user interface to make the game more engaging and user-friendly. Consider incorporating graphics or ASCII art for the hangman figure, adding sound effects, or creating a graphical user interface (GUI).
- **Advanced Gameplay Features**: Introduce advanced gameplay features to make the Hangman game more challenging and interesting. These could include categories or themes for word selection, multiple difficulty levels, or the ability for players to create custom word lists.
- **Scalability**: Discuss how the code can be scaled for larger projects or integrated into a broader gaming application. Address scalability concerns, such as handling a larger word database or supporting multiplayer functionality.
- **Error Handling and Validation**: Explore error handling and input validation techniques to ensure the code gracefully handles unexpected inputs or errors, providing a more robust and user-friendly experience.
- **Testing and Debugging**: Include a section on testing methodologies and debugging techniques used during the development process. Discuss how to identify and address common programming errors.
- **Documentation**: Provide comprehensive documentation for the code, including function descriptions, usage examples, and coding standards. This will make it easier for other developers to understand and use your code.
- **Security Considerations**: Discuss security best practices for handling user input and data, particularly if the game is intended for online or multiplayer use. Address potential vulnerabilities and how to mitigate them.
- **Localization**: Explore the possibility of localizing the game by adding support for multiple

languages. Discuss the challenges and techniques involved in making the game accessible to a global audience.

- **Community and Collaboration**: Consider creating an open-source project for the Hangman game, sharing it on platforms like GitHub, and inviting contributions from the programming community. Discuss collaboration possibilities and the benefits of open-source development.
- **Educational Value**: Highlight the educational value of the Hangman code by discussing how it can be used as a teaching tool for programming concepts, such as loops, arrays, and conditional statements.
- **Feedback and User Experience**: Include a section on user feedback and experiences, if available. Discuss how user feedback was incorporated into the code's development and how it influenced decision-making.
- **Future Roadmap**: Outline a future roadmap for the project, detailing planned enhancements, updates, and improvements. This can demonstrate a commitment to ongoing development and improvement.
- By incorporating these future enhancements into your Hangman code report, you can create a more comprehensive and forward-looking document that showcases not only the current state of the code but also its potential for further development and growth.

Conclusion:

- In conclusion, the development of the Hangman game in C has provided valuable insights into the world of programming, including data structures, user input handling, and game logic. This report has detailed the essential components of the Hangman code, from selecting random words to implementing game mechanics and managing user interactions.
- Throughout the project, we accomplished the following:
- Game Initialization: We successfully initialized the game state, selecting a random word from a predefined list and setting up the initial conditions for gameplay.
- User Interaction: The program effectively handled user input, prompting players to guess letters and providing feedback on their guesses.
- Game Logic: We implemented the core game logic, updating the guessed word and tracking incorrect attempts while ensuring the game's termination conditions were met.
- Game Completion: The Hangman code was able to determine whether the player won or lost the game and provide the appropriate feedback.
- Customization: The code was designed to be easily customizable, allowing for changes in the word list, maximum attempts, and other parameters.
- This project served as a practical exercise in C programming, demonstrating the application of fundamental concepts such as arrays, loops, conditional statements, and functions. It also emphasized the importance of careful design and structured coding practices.
- However, it's worth noting that the presented Hangman code is a basic implementation. There is ample room for improvement and expansion, including enhancing the user interface, incorporating more advanced gameplay features, and implementing data validation to handle invalid inputs gracefully.
- In conclusion, the Hangman code in C has proven to be a valuable learning experience, serving as a foundation for further exploration and refinement in the world of programming. It demonstrates the power of C in creating interactive and engaging applications while highlighting the importance of code organization and clarity.
- This project stands as a testament to the versatility of the C language and its potential for creating entertaining and educational software applications.