TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

PROJECT REPORT
ON
HANGMAN GAME
USING C PROGRAMMING

Under the Supervision of:
Mr. Ganesh Gautam
Department of Electronics and Computer Engineering

Submitted by:	**Submitted to:**
Anuska Bashyal (080BEI007)	Department of Electronics and Computer Engineering
Maunata Pulami (080BEI023)	
Priya Thapa (080BEI030)	
Smriti Khanal (080BEI042)	
Batch: 2080 (I/I), BEI	

Date: April, 2024/ Baisakh, 2081

ACKNOWLEDGEMENT

We are extremely delighted and gratified towards the completion of this project of Computer Programming We are indebted to the entire Pulchowk Campus family and the Department of Electronics and Computer Engineering that provided us with this mini project.

We would like to express our appreciation and special thanks to our mentor Mr. Ganesh Gautam for the theoretical and practical knowledge that he provided on basic programming concepts. The lectures, assignments and guidelines have been of great help for us to successfully complete our project. We appreciate the time and efforts they provided throughout the semester. We cannot forget to mention the lab teachers and the entire department for their persistent and sustained inspiration. Their insightful counsel and recommendations were quite helpful in finishing the assignment. As we have neared the end of our academic semester, this project has not only enhanced our programming skills in C language, but also our teamwork and documentation skills.

Lastly, we want to convey our sincere gratitude towards our friends, classmates and seniors that provided their suggestions and assistance for completion of the project. We would like to sincerely thank everyone who has directly or indirectly contributed to the project.

Thank you all

24th April, 2024

ABSTRACT

The goal of this project is to build a mini game “Hangman” using the C programming language. We have aimed to create a simple game using programming concepts that we learned in class, showcase our proficiency in programming fundamentals, and demonstrate our ability to design an interactive and user-friendly experience.

The Hangman game project exemplifies how the concepts learned in class can be effectively utilized to create functional and engaging software applications, reinforcing the relevance and applicability of our academic coursework in real-life contexts. We extensively employed fundamental concepts such as arrays, loops, file handling, data structures, and external library files to develop a robust and interactive gaming experience.

The report provides a comprehensive overview of the program’s features, design, and functionality, and highlights the key aspects. It is a final project that employs all the concepts of basic programming that we have learnt thus far in this semester as well as many experimental features too.

Keywords: C, C Programming, Education, Entertainment, Game, Hangman,

TABLE OF CONTENTS

1. INTRODUCTION

1.1 The C Programming Language

C is a general-purpose, procedural computer programming language developed by Dennis Ritchie in Bell Laboratories between 1972 and 1973 AD as a successor to Basic Combined Programming Language (BCPL or the B language). C was designed such that code written in C can be translated efficiently into machine level instructions. Code written in C somewhat resembles the English language as it uses keywords such as if, else, for, do, etc. Along with its speed and syntax additional features of C allow it to work at a lower level and thus can bridge the gap between machine level and high-level languages. Due to this, C has found lasting use in systems programming (writing operating systems). It can also be used for applications programming. Applications made in C are generally much faster and more efficient than most other programming languages even with less optimization.

1.2 Background of project

The Hangman game is a classic word-guessing game that has been enjoyed by people for many years. This game is often played by two players, where one provides the word and the other guesses it. This game started a long time ago in Victorian England as a simple game people played in their homes. Over the years, Hangman has grown into a favourite activity for people of all ages, both as fun entertainment and a helpful learning tool.

1.3 Problem Statement

Traditionally, Hangman is seen as a game for entertainment that offers players a fun and engaging way to pass the time. However, its potential as an educational tool is often overlooked. Hangman is a game that not only entertains but also stimulates cognitive abilities such as vocabulary recall and problem-solving. Our project addresses this by transforming the Hangman game into an effective educational tool. This adaptation aims to bridge the gap between leisure and learning, making it possible to acquire knowledge through gameplay.

1.4 Objectives

The primary objectives of this project were:

- To create a Hangman Game using C programming language
- To integrate educational content into the gameplay.

The primary objective of this project is to feature a user-friendly interface displaying the hangman's progress, hints, partially revealed word, guessed letters, and game messages. Key requirements include accessing a diverse word database, implementing core game logic for word selection, limiting number of attempts, tracking entered letters, letter validation, potentially incorporating graphical elements for visual appeal, calculating and managing player scores through file handling, and ensuring robust error handling. We also aim to integrate educational content into the gameplay.

1.5 Project Scope and Applications

The application of this project includes use in classrooms and home-schooling environments to learn vocabulary and terminologies related to a science. It also serves as an entertainment tool for leisure and relaxation and engagement. Various modifications can be made on this project in future enhancing user experience by integrating various educational contents and improving its compatibility with various devices. We can integrate more letter words in the program to make more dynamic and unique in each round.

2. REQUIREMENT ANALYSIS

Hardware/Software Requirements:

For the completion of this project, we used simple DevC++ for writing, debugging and managing the codes implemented in C programming language. Microsoft Word was used

for documentation purposes such as report writing. Required information and references were gathered from reputable sources on the internet, ensuring accuracy and reliability of the project. No special hardware was required besides the computer used for programming purpose for the completion of this project, as it was entirely software-based.

3. THEORY

Header files:

A **C header file** is a text file that contains declarations and definitions of functions, variables, macros, and other constructs that can be shared across multiple source files. The name of a header file ends with the .h extension. It is inserted inside a program by coding the #include preprocessor directive. C header files used in the project are:

1. stdio.h:
   The "Standard Input Output" header file provides functions for input and output operations in C.
   Commonly Used Functions: `printf()`, `scanf()`, `fprintf()`, `fscanf()`, `fgets()`, `fopen()`, `fclose()`.

2. stdlib.h:
   The "Standard Library" header file provides general-purpose functions and macros for memory allocation, process control, conversion, random number generation, and other utilities.
   Commonly Used Functions: `malloc()`, `free()`, `exit()`, `atoi()`, `rand()`, `srand()`, `system()`.

3. string.h:
   The "String" header file provides functions for string manipulation, including operations like copying, concatenating, comparing, and searching strings.
   Commonly Used Functions `strcpy()`, `strcat()`, `strcmp()`, `strlen()

4. ctype.h:
   The "Character Type" header file provides functions and macros for character classification and conversion. It's useful for checking the properties of characters, such as whether they are alphabetic, numeric, uppercase, lowercase, etc.
   Commonly Used Functions: `isalpha()`, `isdigit()`, `isalnum()`, `isupper()`, `islower()`, `toupper()`, `tolower()`.

5. time.h:
   The "Time" header file provides functions and types for date and time manipulation. It includes functions for getting current time, formatting time values, converting between time representations, and performing time arithmetic.
   Commonly Used Functions: `time()`, `ctime()`, `gmtime()`, `localtime()`

User defined function:

A user-defined function is a distinct segment of code crafted by the programmer to execute a particular task. Its purpose is to promote re-usability, improve program clarity and simplify code maintenance. The function entails a declaration outlining its return type, name, and parameters, followed by a body housing executable instruction. When called within the program, the function gets parameter values and executes its body, which may return a value to the calling code. User-defined functions are critical for breaking down complex jobs into smaller chunks.

Syntax:

```
return_type function_name(parameters) {
// Function body (executable statements)
// ...
return value; // (if applicable)
}
```

Types

1. Function with no arguments and no return value
2. Function with no arguments and a return value
3. Function with arguments and no return value
4. Function with arguments and with return value

Pointers:

In C, a pointer is a variable that stores the memory address of another variable. Unlike regular variables that store data directly, pointers store the address of memory locations where data is stored. This capability allows C programmers to work directly with memory, enabling dynamic memory allocation, efficient data manipulation, and advanced programming techniques.

Example:

```
void main(){
int *ptr; // Declares a pointer named ptr that points to an integer
int x = 10;
int *ptr = &x; // Initializes ptr with the address of variable x
printf("Value of x: %d\n", *ptr); //Dereferences pointer using * }
```

Arrays:

In C, an array is a collection of elements of the same data type stored in contiguous memory locations. Arrays provide a way to store multiple values of the same type under a single name, making it easier to work with large sets of data. Each element in an array is accessed using an index, which represents its position within the array.

Syntax: Data_type array_name[size];
Example: int a[10];

Strings:

A string is a sequence of characters terminated by a null character ('\0'). Strings in C are represented using arrays of characters, where each character corresponds to one element in the array. The null character marks the end of the string and is used to indicate where the string ends in memory.

String Declaration:

```
char str1[] = "Hello"; // Static declaration with initialization
```

String handling functions in C are a set of standard library functions provided by <string.h> header. These functions allow you to perform various operations on strings such as copying, concatenating, comparing, and searching. Some of them are:

strcpy() :for copying strings
strcat(): for concatenating strings
strlen(): for getting the length of a string
strcmp() :for comparing string

Looping statements:

Looping statements in C are used to execute a block of code repeatedly if a certain condition is true. There are three main types of loops in C:

1. for loop:
```
for (initialization; condition; update) {
// Code to be executed repeatedly
}
```
2. While loop:
```
while (condition) {
// Code to be executed repeatedly }
```
3. Do while loop:
```
do {
 // Code to be executed repeatedly
} while (condition);
```

Conditional Statements:

1. if statement: Executes a block of code if a specified condition is true.
```
if (condition) {
   // Code to execute if condition is true
}
```
2. if-else statement: Executes one block of code if a condition is true and another block if it is false.
```
if (condition) {
   // Code to execute if condition is true
} else {
   // Code to execute if condition is false
}
```
3. if-else ladder: Allows multiple levels of conditions to be evaluated.
```
if (condition1) {
   // Code to execute if condition1 is true
} else if (condition2) {
   // Code to execute if condition1 is false and condition2 is true
}else {
   // Code to execute if none of the above conditions are true
}
```
4. Switch Statement: Evaluates an expression and executes code blocks based on matching case labels.
   Syntax:
```
switch (expression) {
   case constant1:
      // Code to execute if expression equals constant1
```

```
        break;
    case constant2:
        // Code to execute if expression equals constant2
        break;
    default:
        // Code to execute if no case matches expression
}
```

Structure:

A structure is a composite data type that allows you to group together variables of different data types under a single name. It enables you to create more complex data structures by combining multiple variables into a single unit.

Syntax:

```
struct structure_name {
    // Member variables (also called fields or members)
    data_type1 member1;
    data_type2 member2;
    // ...
};
```

File handling:

File handling in C involves reading from and writing to files using file streams. The <stdio.h> header provides functions and types to perform file input and output operations.

1. Opening a file
   ```
   FILE *fp;
   fp = fopen("filename.txt", "mode");
   ```
2. Reading from file
   ```
   char buffer[255];
   fscanf(fp, "%s", buffer); // Reads a string from the file
   ```
3. Writing to file
   ```
   fprintf(fp, "This is a message.\n"); // Writes a formatted string to the file
   ```
4. Closing file
   ```
   fclose(fp);
   ```

Modes:

Opening Modes	Description
r	Reads an already existing file
w	Open for writing in text mode. If the file exists, its contents are overwritten. If the file doesn't exist, a new file is created.
a	Opens only in the append mode. If the file doesn't exist, a new file is created.
r+	Opens in both reading and writing mode. File must already exist.
w+	Opens in both reading and writing mode. If the file exists, its contents are overwritten. If the file doesn't exist a new file is created.
a+	Opens the file in both reading and append mode. If the file doesn't exist, a new file is created.

4. SYSTEM DESIGNS

4.1 Algorithm

BASIC STRUCTURE of program

1. **Start.**
2. **Accept name.**
3. **Accept letters.**
4. **If letter is right, give 10 points.**
5. **If wrong, -5 points and draw hangman accordingly.**
6. **If all 5 letters guessed, display victory message.**
7. **Else, display losing message.**
8. **Store name and score in file.**
9. **Display leaderboard when prompted by user by accessing data from file.**
10. **Stop.**

MAIN function

1. Start.
2. Include necessary header files.
3. Declare all required functions, global variables, structure
4. Display home page with option for NEW GAME and LEADERBOARD.
5. Accept input ‘choice’ from user
6. IF ‘choice’=1 start new game
    a. Prompt user to enter name
    b. Call function playgame()
7. ELSE IF ‘choice’=2 display leaderboard
    a. Call function leaderboard()
8. Stop.

PLAYGAME () function

1. Start.
2. Load words and hints from file “words.txt”
3. Select random index and ‘word’ using random function.
4. Print welcome message and display hint and hanging stand.
5. For attempts 1 to 4
    a. Accept ‘guess’ character from user.
    b. Check if input is valid (i.e. alphabet)
    c. Check if guess letter already entered.
    d. IF ‘guess’ matches with any letter in ‘word’

        i. Increase 10 points.
        ii. Display correct letter in blank space.
        iii. Attempt not decrease.
    e. ELSE
        i. Decrease 5 points.
        ii. Call function DRAWHANGMAN()
        iii. Decrease available attempts.
6. IF all letters guessed correctly,
   Display victory message, points scored and highest score(call HIGHSCORE function).
7. ELSE Display losing message, required word and current score.
8. Call function SAVESCORE() to save score and name in file.
9. Call function PLAYAGAIN() to ask user if want to play again.
10. IF return TRUE, call PLAYGAME(). //recursive

DRAWHANGMAN() function

1. Draw face, body, hands and legs based on the number of attempts left.

SAVESCORE()

1. Read existing file “highscore.txt” in r mode.
2. Read and save all stored data in array structure ‘u’ until reached EOF.
3. ELSE add name and score to last index ‘i’ of array structure ‘u’.
4. Open file in w more and save the array structure in file.
5. Close file and stop.

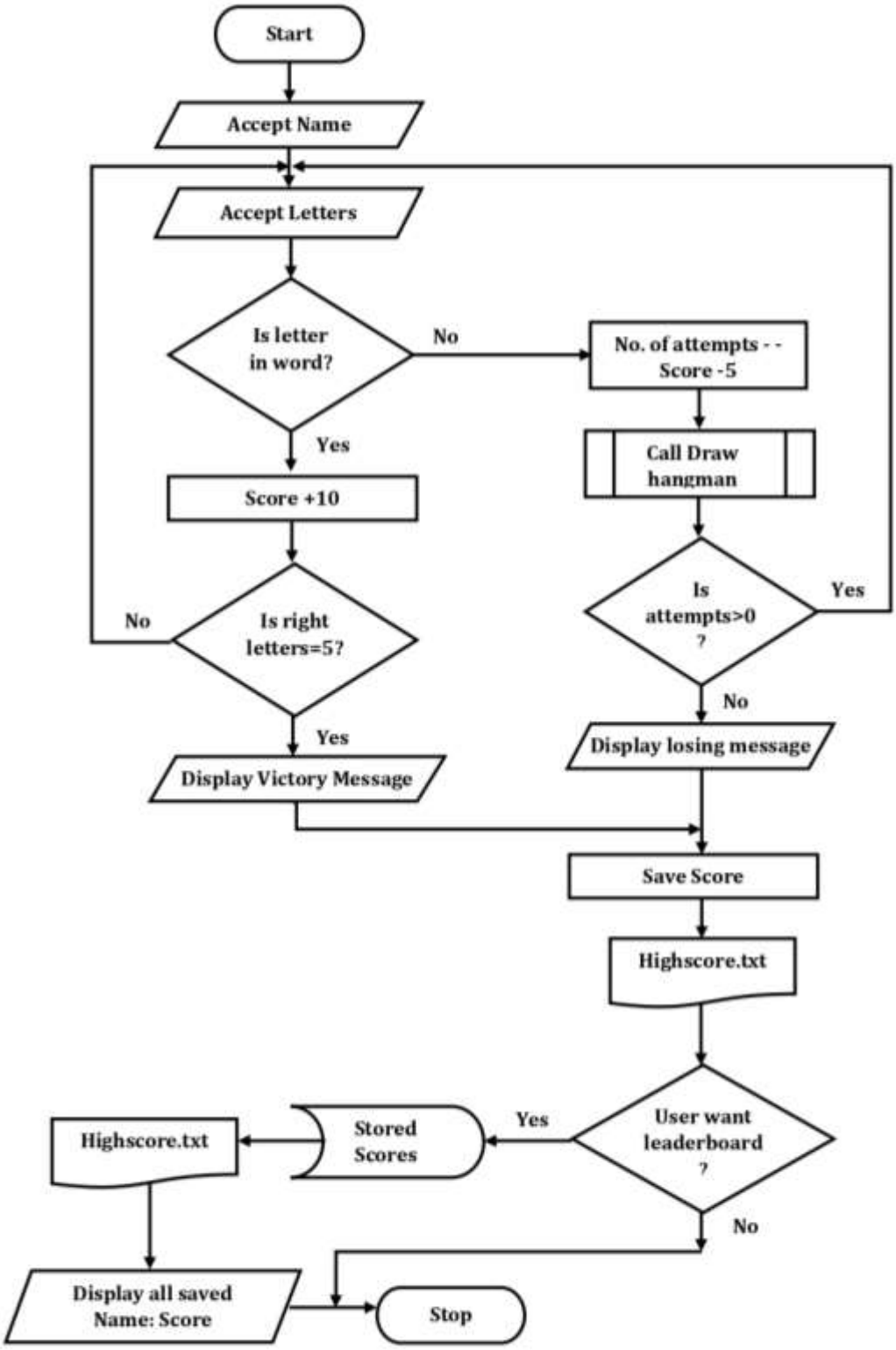
LEADERBOARD() function

1. Read existing file “highscore.txt” in r mode.
2. Read and save all data in array structure ‘u’ until reached EOF.
3. Display data in listed form.
4. Close file and stop.

HIGHSCORE() function

1. Read existing file “highscore.txt” in r mode.
2. IF u.[i]score> than all others, return that value ‘highscore’.
3. Stop.

4.2 Flowchart



5. IMPLEMENTATION

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>

// Function prototypes
void playGame();
void drawHangman(int);
void drawWord(char[], char[]);
int checkGuess(char[], char[], char, int);
void printHint(char[]);
void printWelcomeMessage();
void displayVictoryMessage();
void displayLosingMessage();
int playAgain();
void loadWords(char *filename, char
*words[], char *hints[], int *numWords);
void savescore();
int readHighScore();
void leaderboard();

// Global variable for score
#define MAX_ATTEMPTS 4
int highestScore;
int i,j,k;
int score = 0;

// a struct to store score

struct user {

char name[20];

int score;
}a,u[200],temp;

// Main function
int main() {
printf("    WELCOME TO HANGMAN   \n");

printf("\n\n\n1. Start a New Game\n");
printf("2. Check Leaderboard\n");
printf("Enter your choice: ");

int choice;
scanf("%d", &choice);

switch (choice) {
case 1:
// Start a new game
printf("\nStarting a new game...\n");
printf("\nPlease enter your name (no
spaces, less than 20 chars)\n");
scanf("%s",a.name);
playGame();
break;
case 2:
// Display high score
printf("\nLeaderboard:\n");
leaderboard();
break;
default:
printf("\nInvalid choice. Please select
either 1 or 2.\n");
break;
}
return 0;
}

void playGame(){
// Array of 5-letter words, each with its
associated hint
char *words[100];
char *hints[100];
int numWords;

// Load words and hints from file
loadWords("words.txt", words, hints,
&numWords);

srand(time(NULL)); // Seed the random
number generator

int randomIndex = rand() % numWords;
char word[20];
strcpy(word, words[randomIndex]);

char guessedWord[20];
int len = strlen(word);
for (i = 0; i < len; i++)
guessedWord[i] = '_';
guessedWord[len] = '\0';

char guessedLetters[26];
int attempts = MAX_ATTEMPTS;
int numGuessed = 0;

// Initialize guessed letters array
for (i = 0; i < 26; i++)
guessedLetters[i] = 0;
printWelcomeMessage();
```

```
// Display hint
printHint(hints[randomIndex]);

// Game loop
while (attempts > 0 && numGuessed < len) {
// Draw hangman
drawHangman(attempts);

// Draw word
drawWord(word, guessedWord);

// Print guessed letters
printf("\nGuessed letters: ");
for (i = 0; i < 26; i++) {
if (guessedLetters[i] != 0)
printf("%c ", guessedLetters[i]);
}

// Input guess
printf("\nEnter a letter: ");
char guess;
scanf(" %c", &guess);

// Check if input is a valid letter
if (!isalpha(guess)) {
printf("Invalid input. Please enter a
letter.\n");
continue;
}

// Check if the guess already been made
if (guessedLetters[tolower(guess) - 'a']
!= 0) {
printf("You already guessed that
letter.\n");
continue;
}

// Mark the letter as guessed
guessedLetters[tolower(guess) - 'a'] =
tolower(guess);

// Check if the guess is correct
int right = checkGuess(word, guessedWord,
guess, len);
if (right != 0) {
numGuessed += right;
score += 10 * right;
printf("Good guess!\n");
} else {
attempts--;
score -= 5;
printf("Incorrect guess!\n");
}   }

// Calculate score
if (numGuessed == len) {
displayVictoryMessage();
printf("The word is: %s\n", word);
printf("Your score: %d\n", score);
a.score=score;

highestScore= readHighScore();
if (score > highestScore) {
highestScore = score;
printf("Congratulations! You've achieved a
new highest score!\n");
}
} else {
displayLosingMessage();
drawHangman(attempts);
printf("The word was: %s\n", word);
printf("Your score: %d\n", score);
}

printf("Highest Score: %d\n",
highestScore);

// Play again
if (playAgain()) {
playGame();
}
else
savescore();
}

// Function to draw hangman based on
attempts left
void drawHangman(int attempts) {
switch (attempts) {
case 0:
printf("  _______\n");
printf(" |/     |\n");
printf(" |     (_) \n");
printf(" |     \\|/\n");
printf(" |      |\n");
printf(" |     / \\\n");
printf("_|_\n");
break;
case 1:
printf("  _______\n");
printf(" |/     |\n");
printf(" |     (_) \n");
printf(" |     \\|/\n");
printf(" |      |\n");
printf(" |       \n");
printf("_|_\n");
break;
case 2:
printf("  _______\n");
printf(" |/     |\n");
printf(" |     (_) \n");
printf(" |     \\|/\n");
printf(" |       \n");
```

```
printf(" |\n");
printf("_|_\n");
break;
case 3:
printf("  _______\n");
printf(" |/     |\n");
printf(" |     (_) \n");
printf(" |      | \n");
printf(" |        \n");
printf(" |\n");
printf("_|_\n");
break;
case 4:
printf("  _______\n");
printf(" |/     |\n");
printf(" |       \n");
printf(" |        \n");
printf(" |        \n");
printf(" |\n");
printf("_|_\n");
break;
}  }
// Function to draw the word with
underscores for missing letters
void drawWord(char word[], char
guessedWord[]) {
printf("\n\n");
for (i = 0; i < strlen(word); i++) {
printf("%c ", guessedWord[i]);
}
printf("\n\n");
}

// Function to check if the guessed letter
is correct
int checkGuess(char word[], char
guessedWord[], char guess, int len) {
int found = 0;
for (i = 0; i < len; i++) {
if (word[i] == guess) {
guessedWord[i] = guess;
found ++;
}   }
return found;
}
void printWelcomeMessage() {
printf("Try to guess the word by entering
one letter at a time.\n");
printf("You have %d attempts. Good
luck!\n", MAX_ATTEMPTS);
}

// Function to print hint for the word
void printHint(char hint[]) {
printf("\nHint: %s\n", hint);
}

// Function to display victory message
void displayVictoryMessage() {
printf("\nCongratulations! You guessed the
word!\n");
}

// Function to display losing message
void displayLosingMessage() {
printf("\nGame over! You failed to guess
the word.\n");
}

// Function to ask if player wants to play
again
int playAgain() {
highestScore= readHighScore();
char choice;
printf("\nDo you want to play again?
(Y/N): ");
scanf(" %c", &choice);
if (tolower(choice) == 'y') {
system("cls");
return 1;
} else {
return 0;
}   }
// Function to load words and hints from
file
void loadWords(char *filename, char
*words[], char *hints[], int *numWords) {
FILE *file = fopen(filename, "r");
if (file == NULL) {
printf("Error opening file %s\n",
filename);
exit(1);
}

*numWords = 0;
char line[100];
while (fgets(line, sizeof(line), file)) {
// Remove newline character
line[strcspn(line, "\n")] = 0;

// Split line into word and hint
char *token = strtok(line, ",");
words[*numWords] = strdup(token);
token = strtok(NULL, ",");
hints[*numWords] = strdup(token);

(*numWords)++;
}

fclose(file);
}

//Function to input score into file
void savescore(){
```

```
//reading existing file and saving all
content to ued structure
FILE *file = fopen("highscore.txt", "r");
if (file == NULL) {
printf("Error reading high score
file.\n");
exit(1);
}
i=0;
while(fscanf(file,"%s %d\n
",&u[i].name,&u[i].score)!=EOF){
i++;
}
fclose(file);
//save recent data in last place of
unsorted
file = fopen("highscore.txt", "w");
strcpy(u[i].name,a.name);
u[i].score=a.score;
for (j=0;j<=i;j++){
fprintf(file,"%s
%d\n",u[j].name,u[j].score);
}
fclose(file);
}

void leaderboard(){
//reading existing leaderboard file
FILE *file = fopen("highscore.txt", "r");
if (file == NULL) {
printf("Error reading high score
file.\n");
exit(1);
}
i=0;
while(fscanf(file,"%s %d\n
",&u[i].name,&u[i].score)!=EOF){
printf("%10s :
%5d\n",u[i].name,u[i].score);
i++;
}
fclose(file);
}

// Function to read the high score from
the file
int readHighScore() {
FILE *file = fopen("highscore.txt", "r");
if (file == NULL) {
printf("Error opening high score
file.\n");
return 0;
}
i=0;
while(fscanf(file,"%s %d\n
",&u[i].name,&u[i].score)!=EOF){
i++;
}
int highScore=u[0].score;
for (j=1;j<=i;j++){
if(highScore<u[i].score)
highScore=u[i].score;
}
return highScore;
fclose(file);
}
```

6. RESULTS AND DISCUSSION

This project resulted in a fully functional game that successfully achieved its objectives of providing entertainment and education to the users. Players were able to learn about various scientific concepts while enjoying the classic Hangman gameplay.

1. Before starting, home page is displayed and the program asks user to input number.

```
                    WELCOME TO HANGMAN



1. Start a New Game
2. Check Leaderboard
Enter your choice:
```

2. It prompts user to enter their name.

```
1. Start a New Game
2. Check Leaderboard
Enter your choice: 1

Starting a new game...

Please enter your name (no spaces, less than 20 chars)
```

3. After entering name, it displays the game.

```
Please enter your name (no spaces, less than 20 chars)
Player
Try to guess the word by entering one letter at a time.
You have 4 attempts. Good luck!

Hint:  any substance having ph more than 7.
  _______
 |/      |
 |
 |
 |
 |
_|_


_ _ _ _ _


Guessed letters:
Enter a letter:
```

4. If correct letter is guessed, it fills in the blanks.

```
Enter a letter: b
Good guess!
  _______
 |/      |
 |
 |
 |
 |
_|_


b _ _ _ _


Guessed letters: b
Enter a letter:
```

5. If wrong letter, following output is seen.

```
Guessed letters: b
Enter a letter: l
Incorrect guess!
  _______
 |/      |
 |      (_)
 |       |
 |
 |
_|_


b _ _ _ _


Guessed letters: b l
Enter a letter:
```

6. After successful completion of word,

```
  _______
 |/      |
 |
 |
 |
 |
_|_


l a s e _


Guessed letters: a e l s
Enter a letter: r
Good guess!

Congratulations! You guessed the word!
The word is: laser
Your score: 50
Congratulations! You've achieved a new highest score!
Highest Score: 50

Do you want to play again? (Y/N):
```

7. If input letters are wrong 4 times

```
Guessed letters: i o q
Enter a letter: r
Incorrect guess!

Game over! You failed to guess the word.
  _______
 |/      |
 |      (_)
 |      \|/
 |       |
 |      / \
_|_
The word was: metal
Your score: 30
Highest Score: 0

Do you want to play again? (Y/N):
```

8. When pressed Y in play again

```
Try to guess the word by entering one letter at a time.
You have 4 attempts. Good luck!

Hint:  electropositive elements.
  _______
 |/      |
 |
 |
 |
 |
_|_


_ _ _ _ _


Guessed letters:
Enter a letter:
```

9. When pressed N in play again

```
Do you want to play again? (Y/N): n

Process returned 0 (0x0)   execution time : 20.832 s
Press any key to continue.
```

10. when pressed 2 in home screen, displays all saved scores

```
1. Start a New Game
2. Check Leaderboard
Enter your choice: 2

Leaderboard:
    Player :    50

Process returned 0 (0x0)   execution time : 2.356 s
Press any key to continue.
```

7. CONCLUSION

In conclusion, the development of the Hangman project resulted in an engaging blend of entertainment and educational gaming experience. Through the utilization of the C programming language, meticulous planning, and research, we have seamlessly integrated educational content into the gameplay, allowing players to expand their knowledge of scientific concepts while engaging in the classic Hangman challenge.

Thorough testing and documentation have ensured the quality and reliability of the game, setting a solid foundation for its deployment and future enhancements. The user interface design, though primarily text-based, ensures accessibility and ease of use for players of all ages. As the game goes live, we remain committed to its ongoing improvement and expansion.

The Hangman game project has been an enriching educational endeavor that allowed us to apply various concepts of C programming in a real-life context. Through the development of this game, we deepened our understanding of fundamental programming principles such as loops, file handling, and data structures, while also gaining practical experience in user interface design and error handling.

The project provided an opportunity to bridge theoretical knowledge with practical application, reinforcing our programming skills and problem-solving abilities. The Hangman game project has enriched our learning journey by fostering creativity, collaboration, and critical thinking. As we progress in our academic journey, the skills acquired from this project will serve as a solid foundation for tackling more complex challenges in computer science.

8. LIMITATION AND FUTURE RECOMMENDATION

Although the C programming based hangman game is useful and complies with the criteria, there are some project constraints that were faced during development.

1. Graphics: Advanced graphics is not integrated into the program, the developers not being completely familiar to the new concept. The DevC++ output interface is not the ideal one for good graphics work, and so we have displayed hangman using only special characters. In future, we can integrate IDE that supports graphic to develop a more robust game.
2. Language constraints: As C is a beginner level language, there are limited features and more features can be integrated when used some other programming language which is more advanced and deals with object.
3. Time constraints: The project development was limited by time constraints. This has limited the depth and scope of the project.
4. Proper Analysis: A proper analysis is to be conducted based upon the user's level of knowledge.

9. REFERENCES

- W3 Schools https://www.w3schools.com/c/
- https://www.programiz.com/c-programming
- Stack Overflow, https://stackoverflow.com/
- https://github.com/tintinmj/Hangman-game/blob/master/game.c
- Chat GPT, https://chat.openai.com/