NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

Report on VocabQuest

Word Guessing Game

Team Members:

Haseeb Yaqoob *CT-172*
Ahmed Raza *CT-171*
M.Mughees Sabtain *CT-195*

Course Code: CT-175
Course Title: Programming Fundamentals

Presented to:
Sir Wajihuddin

Submission Date: **26/11/2024**

Contents

Introduction

For a long time, word games have been source of entertainment and mental challenge for people. With the advancement of technology, there is an increase in demand in players' expectation as they ask for more challenging and more dynamic experiences.

VocabQuest, an interactive vocabulary game was developed using the C programming language as an inspiration from Wordle. This game helps users to improve their vocabulary and thinking abilities by allowing them to guess secret words in a limited number of attempts along with feedback to guide them as they play.

How It Works:

Players get six chances to guess each word and completing a level unlocks the next one. The game uses code to check guess, compare it to the secret word, and provide feedback. It also checks for the input to make sure players only enter proper words. The use of colored text makes the game fun allowing users to interact with it.

VocabQuest has three levels. The first level challenges players to guess 5-letter fruit names, the second focuses on 6-letter country names, and the third level introduces 7-letter random words to really test your vocabulary skills.

This report explains how VocabQuest was created, the challenges faced during development, and what's next for the game. It's not just a game but a fun way to build your vocabulary and test your word skills

Tools That Made VocabQuest Possible

The development of the VocabQuest involves various technical tools to ensure an efficient, and user-friendly application.

Programming Language:
C Language: The core of the system is implemented in C. Chosen for its performance and control over system resources, C allows for the creation of a lightweight and efficient application.

Development Environment:
Integrated Development Environment (IDE): Visual Studio Code was used for writing, debugging, and testing the C code. These tools provide features like syntax highlighting, code completion and implementation of libraries to showcase the graphics of the game.

Libraries:
Certain standard C libraries were used in the system:

string.h for string manipulation.
stdio.h for input/output operations.
stdlib.h for dynamic memory allocation.
ctype.h to check if characters are alphabetic and convert them to uppercase.

Version Control:
Git: Git was used for version control to track changes in the code, collaborate with other members, and maintain different versions of the project.

3. Features in Detail with Code and Algorithm

The VocabQuest game includes multiple levels where the user guesses words based on a given number of attempts. Each level presents a set of words with varying lengths, and the game continues until the player either guesses the word correctly or runs out of attempts.

3.1. Adding New Words to the Game and generating Feedback

Description: This feature enables the system to provide feedback on a user's guessed word compared to a secret target word in a word-guessing game.

Algorithm:

1) Input Parameters:

The function accepts three inputs:

guess: The user's guessed word.
secret: The target word to be guessed.
word_length: The length of the words (both guess and secret).

2) Initialize Feedback:

- A feedback string is created with the same length as the secret word, initialized with underscores ('_') to represent unmatched letters.
- The system loops through each character of the guess. If the character matches the corresponding character in the secret word (same position), it is marked as 'G'.
- For each unmatched character in the guess, the system checks if it exists elsewhere in the secret word but at a different position. If found, it is marked as 'Y'.

3) Output Feedback:

The function prints the feedback string, where:

- 'G' is displayed in green to indicate a correct letter in the correct position.
- 'Y' is displayed in yellow for a correct letter in the wrong position.
- '_' is displayed in red to indicate incorrect letters.

This approach ensures a clear and visually distinct feedback mechanism for the user during gameplay.

Code:

```
void print_feedback(const char *guess, const char *secret, int word_length) {
    char feedback[word_length + 1];
    int i, j;
    for (i = 0; i < word_length; i++) {
        if (guess[i] == secret[i]) {
            feedback[i] = 'G';
        } else {
            feedback[i] = '_';
        }
    }
    for (i = 0; i < word_length; i++) {
        if (feedback[i] == '_') {
            for (j = 0; j < word_length; j++) {
                if (guess[i] == secret[j] && feedback[j] != 'G') {
                    feedback[i] = 'Y';
                    break;
```

```
            }
            }
        }
    }

    printf("Feedback: ");
    for (i = 0; i < word_length; i++) {
        if (feedback[i] == 'G') {
            printf(GREEN "%c " RESET, guess[i]);
        } else if (feedback[i] == 'Y') {
            printf(YELLOW "%c " RESET, guess[i]);
        } else {
            printf(RED "_ " RESET);
        }
    }
    printf("\n");
}
```

3.2. Displaying Instructions of the Game

Description: This function displays the instructions for the VocabQuest game, customized based on the level provided, explaining the rules of the game.

Algorithm:

1. Input:

level: An integer representing the current level of the game.

2. Steps:

- Check if the current level is 0 (for the first level of the game).
- Display the game's general rules, including the number of attempts and word length.
- Ask the user to press Enter to start the game**.**

Code:

```
void display_instructions(int level) {
    if (level == 0) {
        printf(MAGENTA "\n===== How to Play Wordle - Level %d =====\n" RESET, level + 1);
        printf("Guess the Wordle in %d tries.\n\n", MAX_ATTEMPTS);
        printf(" - Each guess must be a valid %d-letter word.\n", level_lengths[level]);
        printf(" - The color of the tiles will change to show how close your guess was to the
        word.\n\n");
        printf("Examples:\n");

        printf(GREEN " W " RESET " O R D Y\n");
        printf("W is in the word and in the correct spot.\n\n");

        printf("L " YELLOW " I " RESET " G H T\n");
        printf("I is in the word but in the wrong spot.\n\n");
        printf("R O G " RED " U " RESET " E\n");
        printf("U is not in the word in any spot.\n\n");

        printf("Press Enter to start the game...\n");
        while (getchar() != '\n');
    }
}
```

3.3. Input Validation Function

Description: : This function validates the user's input to ensure that it follows the game's rules. It checks whether the guessed word has the correct length and consists only of alphabetic characters (A-Z, a-z).

Algorithm:

1. Steps:

- Compare the length of the input with the expected word_length.
- If the lengths do not match, return 0 indicating invalid input.
- Loop through each character in the input string.
- Use the isalpha() function to check if each character is alphabetic.
- If any character is not an alphabetic letter, return 0.
- If the input meets both conditions (correct length and all alphabetic characters), return 1 indicating valid input.

2. Output:

- Returns 1 if the input is valid (correct length and contains only alphabetic characters).
- Returns 0 if the input is invalid (incorrect length or contains non-alphabetic characters).

Code:

```
int is_valid_input(const char *input, int word_length) {
    if (strlen(input) != word_length) return 0;
    for (int i = 0; i < word_length; i++) {
        if (!isalpha(input[i])) {
            return 0;
        }
    }
          return 1;
}
```

3.4. Game Start Function

Description: The start_game function initiates the word guessing game for a given level, where the user tries to guess a secret word within a set number of attempts

Algorithm:

1. **Randomly Select a Secret Word:**
   - A random number is generated using rand() to select a secret word from the list of words associated with the given level.
   - The length of the word is also retrieved based on the current level.
2. **Game Introduction:**
   - Print a welcome message indicating the current level and the number of attempts allowed.
3. **Main Game Loop:**
   - The player is given up to MAX_ATTEMPTS to guess the secret word. For each attempt: the player is asked to input a guess.

Input Validation:

   - The input is validated to ensure it is a valid word of the correct length using the is_valid_input function. If invalid, the player is informed and prompted to try again.

Feedback Generation:

   - The print_feedback function is called to show how close the player's guess is to the secret word.

Check for Correct Guess:

   - If the guess matches the secret word, the player is congratulated, and the game ends.
   - If the guess is incorrect and the player still has attempts remaining, the loop continues.

Code:

```
void start_game(int level) {
    srand(time(NULL));
    const char *secret = level_words[level][rand() % level_sizes[level]];
    int word_length = level_lengths[level];
    char guess[word_length + 1];
    int attempts = 0;

    printf(MAGENTA "Welcome to Wordle - Level %d!\n" RESET, level + 1);
    printf(BLUE "Guess the %d-letter word. You have %d attempts.\n" RESET, word_length,
    MAX_ATTEMPTS);

    while (attempts < MAX_ATTEMPTS) {
        printf("Attempt %d: ", attempts + 1);

        fgets(guess, word_length + 2, stdin);
        guess[strcspn(guess, "\n")] = '\0';

        if (!is_valid_input(guess, word_length)) {
            printf(RED "Invalid input. Please enter a valid %d-letter word.\n" RESET, word_length);
            continue;
        }
```

```
        print_feedback(guess, secret, word_length);
        attempts++;
        if (strcmp(guess, secret) == 0) {
            printf(GREEN "Congratulations! You've guessed the word: %s\n" RESET, secret);
            return;
        }
    }
    printf(RED "Sorry! You've used all attempts. The word was: %s\n" RESET, secret);
}
}
}
```

3.5. Main Function for VocabQuest Game

Description: The main function serves as the entry point for the VocabQuest game. It manages the game's flow, handles the user's decision to play or quit, and controls the progression through the levels. After the game starts, the player is asked to guess the word at each level, with instructions displayed and feedback given for each attempt. Upon completing a level or all levels, the player is notified of their progress or game completion.

Algorithm:

1. Input:

- User's input choice (choice) to either play or quit the game.

2.Display Welcome Message:

- The game starts with a welcome message.
- The user is prompted to either press Enter to play or type 'Q' to quit.

3.Quit Game Check:

- If the user presses 'Q' or 'q', the game exits with a goodbye message.
- If any other input is given, the newline characters are flushed to avoid interference with the gameplay.

4.Game Levels Loop:

- The game proceeds with levels, starting from level 0.

  For each level:
- The display_instructions function is called to show the player how to play the game at the current level.
- The start_game function is called to allow the player to guess the secret word.
- After completing the level, the player is congratulated and progresses to the next level.

5.End of Game:

- If all levels are completed, a completion message is shown, thanking the player for playing.

Code:

```
int main() {
    char choice;
    int level = 0;
    printf(MAGENTA "===== Welcome to Wordle =====\n" RESET);
    printf("Press Enter to play or type 'Q' to quit: ");
    choice = getchar();
    if (choice == 'Q' || choice == 'q') {
        printf(RED "Exiting the game. Goodbye!\n" RESET);
        return 0;
    }
    if (choice != '\n') {
        while (getchar() != '\n');
    }
    while (level < LEVEL_COUNT) {
        display_instructions(level);
        start_game(level);
        level++;        if (level < LEVEL_COUNT) {
            printf(MAGENTA "Congratulations! You've advanced to level %d.\n" RESET, level +
1);
        } }
    printf("You've completed all levels. Thanks for playing!\n");
    return 0;}
```

4. Methodology

The methodology section outlines the approach adopted to develop and implement the VocabQuest Game. The methodology covers various phases: requirement analysis, design, development, testing, and deployment.

4.1. Requirement Analysis

Objective: To gather and analyze the requirements for the VocabQuest game.

Activities:

- Survey Distribution: Took opinion and surveys from students to collect data on their needs, preferred game features, and feedback on existing similar games.
- Online Research: We researched existing vocabulary-building games or educational apps online to identify popular features, user complaints, and best practices in game design.

4.2 System Design

Objective: To create a blueprint for the system that meets the specified requirements.

Activities:

- **Game Logic and Flow Design:** We looked over the game mechanics, including word validation, feedback generation (e.g., green, yellow, red indicators for correct or incorrect guesses), and handling different levels of difficulty..
- **User Interface Design:** Designed a user-friendly UI that ensures ease of navigation, smooth game flow, and visually engaging elements for feedback (using color coding and clear instructions).

Outcome: A fully functional game that meets all requirements and can be played across various levels with engaging word-guessing challenges.

4.3. Development

Objective: To implement the VocabQuest game based on the collected data.

Activities:

- **Environment Setup:** Set up the development environment with necessary tools and libraries.
- **Coding:** Developed the system using C language.
- **Database Integration:** Implement the database to store and retrieve data for word lists, user progress, and game statistics
- **Version Control:** Used Github to manage code changes and collaborate with team members.

Outcome: A fully functional game that meets all requirements and can be played across various levels with engaging word-guessing challenges.

4.4. Testing

Objective: To validate the game's functionality, user experience, and overall performance.

Activities:

Tested individual components such as word validation, feed back mechanism to ensure they work as expected.

By doing this we ensured: A stable, bug-free game that delivers a smooth user experience across various devices and platforms.

5. Challenges Faced during Development of VocabQuest

5.1. Invalid Input Handling

We faced issues when players entered guesses with incorrect lengths or non-alphabetic characters. To fix this, we implemented an input validation function that ensures guesses are valid before proceeding.

```
if (!is_valid_input(guess, word_length)) {
    printf(RED "Invalid input! Enter a valid %d-letter word.\n\n" RESET, word_length);
    continue;
}
```

5.2. Visual Feedback

Players found the feedback hard to interpret initially. To solve this, we used colored output for correct letters (green), misplaced letters (yellow), and incorrect letters (red underscores), enhancing user understanding of the feedback.

```
if (feedback[i] == 'G') {
    printf(GREEN "%c " RESET, guess[i]);
}
else if (feedback[i] == 'Y') {
    printf(YELLOW "%c " RESET, guess[i]);
}
else {
    printf(RED "_ " RESET);
}
```

5.3. Game Over Condition:

A bug allowed players to continue guessing after running out of attempts. We implemented a proper game-over condition to terminate the level when players have used all of their attempts.

```
if (attempts >= MAX_ATTEMPTS) {
    printf(RED "Out of attempts! The word was: %s\n\n" RESET, secret);
    break;
}
```

6. Limitations of VocabQuest

Absence of Time-Limited Challenges

The game lacks a timer, so players can take unlimited time to guess the word. Adding time-based challenges could increase the difficulty and excitement for advanced players.

No Multi-Player or Competitive Mode

VocabQuest is designed for single-player gameplay only. Introducing a multiplayer mode, such as turn-based guessing or timed challenges, could make the game more engaging and competitive.

No Penalty for Invalid Input

While invalid input is handled gracefully, players face no penalties for repeatedly entering incorrect or invalid guesses. Introducing a small penalty (e.g., reducing attempts) could encourage more thoughtful guesses.

No Save or Progress Tracking

Players must complete all levels in one session, as there is no save or progress tracking system. Adding a save feature or progress indicator would allow players to resume their game later and track their achievements.

7. Gameplay Screenshots

```
========== Welcome to VOCAB QUEST ==========
Press Enter to play or type 'Q' to quit:

===== How to Play Wordle - Level 1 =====
Guess the Wordle in 6 tries.

 - Each guess must be a valid 5-letter word.
 - The color of the tiles will change to show how close your guess was to the word.
 - Press 'H' to get a hint.

Examples:
 W  O R D Y
W is in the word and in the correct spot.

L  I  G H T
I is in the word but in the wrong spot.

R O G  U  E
U is not in the word in any spot.

Press Enter to start the game...

Welcome to VOCAB QUEST - Level 1!
Guess the 5-letter word. You have 6 attempts.

Attempt 1: apple

Feedback: a _ _ _ _

Attempt 2: peach

Feedback: _ _ a _ _

Attempt 3: lemon

Feedback: _ _ _ _ _

Attempt 4: mango

Feedback: _ a _ g _

Attempt 5: guava

Feedback: g u a v a

Congratulations! You've guessed the word: guava


Congratulations! Proceeding to Level 2...
```

Gameplay Screenshots

```
Congratulations! Proceeding to Level 2...


Welcome to VOCAB QUEST - Level 2!
Guess the 6-letter word. You have 6 attempts.

Attempt 1: pakistan
Invalid input! Enter a valid 6-letter word.

Attempt 1: Invalid input! Enter a valid 6-letter word.

Attempt 1: denmark
Invalid input! Enter a valid 6-letter word.

Attempt 1: Invalid input! Enter a valid 6-letter word.

Attempt 1: france

Feedback: _ _ a _ _ _

Attempt 2: cyprus

Feedback: _ _ _ _ u _

Attempt 3: kuwait

Feedback: k u w a i t

Congratulations! You've guessed the word: kuwait
```

```
Congratulations! Proceeding to Level 3...


Welcome to VOCAB QUEST - Level 3!
Guess the 7-letter word. You have 6 attempts.

Attempt 1: concert

Feedback: c _ n _ e _ t

Attempt 2: picture

Feedback: _ i c t _ _ e

Attempt 3: quarter

Feedback: _ _ a _ t e _

Attempt 4: cabinet

Feedback: c a b i n e t

Congratulations! You've guessed the word: cabinet

You've completed all levels. Thank you for playing VOCAB QUEST!
```

8. Future Scope

VocabQuest has the potential to evolve into an even more engaging and comprehensive language-learning tool by incorporating the following features:

8.1. AI-Powered Word Game

- **Description**: Use artificial intelligence to adapt the game's difficulty, vocabulary suggestions, and challenge levels based on individual player progress and preferences

8.2. Multiplayer Mode

- **Description**: Introduce competitive and collaborative multiplayer modes where players can challenge friends or work together to solve word puzzles.

8.3. Voice Interaction Features

- **Description**: Implement speech recognition to allow players to practice pronunciation and interact with the game using voice commands.

8.4. Enhanced Vocabulary Challenges

- **Description**: Add more diverse challenges, such as sentence-building, word association, and idiomatic expressions, to deepen language skills.

9. Contributions of Group Members

Name	Role	Contributions
Haseeb Yaqoob (CT-172)	Data Management and Gameplay Mechanics	- Categorized and stored words by length (5, 6, or 7 letters). - Implemented tracking of incorrect guesses with error messages.
Ahmed Raza (CT-171)	Documentation and Gameplay Mechanics	- Added new words to the game. - Generated feedback logic. - Prepared a detailed report on game features, methodology, and challenges.
M. Mughees Sabtain (CT-195)	Game Logic Development	- Developed word selection based on difficulty level. - Implemented input validation using `ctype.h`. - Managed dynamic memory allocation with `stdlib.h`.

Conclusion

The development and implementation of VocabQuest, a word-guessing game built using the C programming language, has resulted in an engaging game

The game includes three difficulty levels and features such as word categories, hints, input checking, and instant feedback, making it easy and enjoyable to use.

The project successfully achieved its goal of creating a platform that helps players improve their vocabulary while having fun

There is still room to make the game better. Future updates could add a colorful user interface, a multiplayer mode for playing with friends, and involving an AI to completely change the way we play games.

In conclusion, this project is a strong starting point for creating more educational games. It shows how technology can make learning enjoyable and interactive. With further improvements and teamwork between developers, educators, and players, tools like VocabQuest can become even more useful and popular.

References:

1. Github
2. Youtube
3. Wordle