The Ultimate Quiz Game

Project Report

Project Title: The Ultimate Quiz Game
Course/Subject: — C Programming Language
Submitted By: —Adhya Agarwal
Submitted To: — Mrs. Dolly Das
Date: —28-Dec-25

2. Abstract

This project presents a console-based quiz application developed in the C programming language. The system features multiple quiz levels, lifelines, a high-score mechanism, and time-based answering. Questions are divided in five diverse categories , promoting user engagement and knowledge assessment. The program demonstrates structured programming concepts, file handling, randomization, and modularity. This report details the problem definition, algorithmic design, implementation approach, testing methods, results, and potential areas for future enhancement.

3. Problem Definition

The objective of this project is to design and implement an interactive quiz game that:

- Presents a structured set of questions across multiple levels.
- Allows the user to answer within a specified time limit.
- Provides lifelines to assist gameplay.
- Maintains and updates a persistent high score.
- The system must be simple, user-friendly, and capable of evaluating the user's knowledge across multiple domains.

4. System Design (Algorithms)

Main Program Algorithm

1. Start program.
2. Initialize total score and load existing high score.
3. Display welcome banner.
4. For each of the five quiz levels:

   Reset lifelines.

   - Display level name.
   - For each of the 10 questions:
     - Display question and randomized options.
     - Ask user if they want to use a lifeline.
     - Process selected lifeline.
     - Begin timer and accept answer.
     - Update score based on correctness and lifeline usage.
   - Add level score to total score.
5. Display final total score and appreciation message.
6. Compare score with stored high score and update if necessary.
7. End program.

`ask()` Function Algorithm

1. Shuffle the order of the four answer options.
2. Display question and options.
3. Ask user about lifeline use.
4. If lifeline chosen:
   - Process Skip, Fifty-Fifty, or Double Points logic.
5. Start the timer and accept answer.
6. Validate answer and check for time expiration.
7. Return appropriate score value.

5. Implementation Details (with snippets)

Example: Question Structure

```
struct Question {
    char q[256];
    char opt[4][128];
    int correct;
};
```

Loading High Score

```
int loadHighscore(){
    FILE *f=fopen("highscore.txt","r");
    if(!f)return 0;
    int s; fscanf(f,"%d",&s); fclose(f); return s;
}
```

Asking a Question

```
printf("%s\n", q->q);
    for(int i = 0; i < 4; i++)
        printf("%d. %s\n", i+1, q->opt[order[i]]);
```

Applying Fifty-Fifty Lifeline

```
int wrongs[3], idx = 0;
            for(int i = 0; i < 4; i++) if(order[i] != q->correct)
wrongs[idx++] = i;
            int keep = wrongs[rand() % 3];
            printf("Fifty-Fifty applied:\n");
            printf("1. %s\n", q->opt[order[q->correct]]);
            printf("2. %s\n", q->opt[order[keep]]);
```

6. Testing & Results

Testing Approach

- **Functional testing** for question display, lifeline behavior, scoring logic, and time limits.
- **Boundary testing** for lifeline usage (ensuring each can be used once per level).
- **File testing** to verify high-score saving and retrieval.
- **Input testing** with valid and invalid answers.

Results Summary

- All five levels executed successfully.
- Lifelines operated correctly and only once per level.
- Timer correctly ended the game after exceeding 90 seconds.
- High-score file updated accurately upon achieving a new record.
- No crashes or memory issues observed during extended testing.

7. Conclusion & Future Work

Conclusion

The Ultimate Quiz Game effectively demonstrates the use of C programming concepts including control structures, arrays, structures, functions, file handling, and randomization. The game offers an engaging quiz experience while maintaining structural clarity and modular organization.

Future Work

- Adding GUI or graphical interface.
- Including more question categories.
- Allowing difficulty levels or adaptive questioning.
- Adding sound effects and color-coded output for enhanced user experience.

8. References

- Kernighan, B. W., & Ritchie, D. M. *The C Programming Language*.
- Online C documentation and reference materials.
- General programming resources for struct handling, file I/O, and time functions.
- https://youtu.be/HDlm1imM7mU?si=VccgXVbyEL_TuW-M