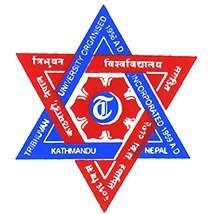
****

**TRIBHUWAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**PURWANCHAL CAMPUS**

**DHARAN**

**QUIZ C**

**A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS**

**AND COMPUTER ENGINEERING IN THE PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE PARTICAL COURSE ON**

**C-PROGRAMMING**

**Submitted By:**

**Mr . Anuj Kumar Thakur (PUR081BCT006)**

**Mr.Bikash Bist (PUR081BCT012)**

**Mr.Jigyasa Bhattarai (PUR081BCT024)**

**Mr. Aaditya Kumar Karn (PUR081BCT001)**

**Submitted to:**

**Department of Electronics And Computer Engineering , Purwanchal Campus , Dharan**

**Institute of Engineering, Tribhuvan University**

**Dharan, Nepal [Falgun 27, 2081]**

1

**Acknowledgment**

We would like to express our sincere gratitude to Mr. Mandip Rai and **ACES** members for their invaluable guidance and support throughout our C- programming project. Their insightful suggestions, constructive feedback, and unwavering encouragement were instrumental in our project. Their expertise and mentorship have been truly inspiring, and we are grateful for his contribution to our learning journey.

**2**

**Contents**

1. Introduction 4

* Objective……………………………………………………………………………. 4
* Features of game………………………………………………………………… 4

1. System , Libraries and Function 5
2. Methodology 6

* Development tools…………………………………………………………. .6
* Development Process……………………………………………………… 6-9

4. Project Scope 10

5.Project Schedule 11

* Timeline……………………………………………………………………………11

3

**Introduction**

We propose to develop an interactive quiz game using C-programming. The primary purpose of this quiz game is to provide an **engaging and educational platform** where players can test and improve their knowledge in a fun and interactive way. We have include several features to make it more engaging such as multiple-choice question, random questions where questions are shuffled to give players a unique experiences, prize money and life line features makes it more interesting.

**Objectives**

1. To create a quiz game that makes learning interesting and fun.

2. For practical application of C- Programming concept.

* File handling
* System function
* User input handling etc.

1. To provide instant feedback and results to users on their quiz performance.
2. To enhancing quick thinking by enabling timer based answering system.

**Features of game:**The quiz game has several engaging features designed to provide an interactive and challenging experience. Questions are loaded from an external file (questions.txt), with multiple-choice options and predefined correct answers. Each question is associated with a 30-second timer, adding pressure to answer quickly, and if time runs out, the game ends. The game also includes lifelines such as 50/50, which removes two incorrect options, and Skip Question, allowing players to bypass difficult questions. A prize money system is in place, with increasing amounts based on question difficulty. The game uses a color-coded interface for better readability, enhancing the user experience. Players progress through the game by answering questions correctly, earning prize money with each right answer. If a player answers incorrectly or times out, the game ends. Additionally, the questions are shuffled randomly, ensuring a unique experience each time. Finally, the game tracks results by saving the player's earnings to a file (results.txt), allowing for performance tracking across sessions.

4

**System, Libraries and function**

The quiz game system is built as a **console-based application** using the **C programming language**. It is designed to run on the **Windows operating system**, utilizing Windows-specific libraries for system-level operations such as console output and timer management. The game operates with a **text-based interface**, where players interact through the keyboard and console display.

The system uses several standard libraries: **stdio.h** for input and output operations, **stdlib.h** for memory allocation and random number generation, **string.h** for string manipulation, **ctype.h** for character handling (e.g., converting user input to uppercase), and **time.h** for generating randomization based on system time. The **windows.h** library is used for handling timer queues, setting console text colors, and other Windows-specific functionalities, ensuring smooth operation and enhanced user experience.

Functions:

In this quiz game file handling is essential for loading questions and saving results. The fopen() function opens questions.txt in read mode ("r") to load quiz questions into a struct array, using fscanf() to parse each question’s text, options (A-D), correct answer, and timeout duration. After the game ends, fopen() with append mode ("a") ensures new player results are added to results.txt without overwriting existing data, with fprintf() writing the player’s name and earnings. Files are properly closed with fclose() to prevent leaks. The fgets() function captures the player’s name from standard input (stdin), sanitizing it by removing the newline character. Beyond file handling, key functions include play\_game(), which manages the quiz flow, timers, and lifelines like "Fifty-Fifty" (hiding two wrong options via use\_lifeline()) and "Skip Question." The shuffle\_questions() function randomizes the question order using the Fisher-Yates algorithm, while start\_timer() and stop\_timer() leverage Windows API for a 30-second countdown. Results are displayed with colorful console output using printf() and ANSI escape codes. Dynamic memory allocation (malloc()) initializes the question array, ensuring scalability if the question pool grows. Together, these functions create an interactive experience, blending file I/O, user input, and real-time game login.

5

**Methodology**

**Development Tools**

* Programming Language: C
* IDE: Visual Studio Code
* Libraries/frameworks: on need

**Development Process**

**1. Planning the Structure:**

* **Game Flow**: The first step in development is deciding the core flow of the game. This involves understanding how the player progresses through questions, how the lifelines will work, and what happens at the end of the game. Key points include:
  + The player answers questions sequentially.
  + A timer counts down for each question.
  + If the player answers wrong or times out, the game ends.
  + After the game ends, the player's score is displayed and saved.
* **Question Structure**: Deciding how to structure questions and answers is crucial. Each question includes:
  + The text of the question.
  + Four answer options (A, B, C, D).
  + The correct answer.
  + A timeout value (in seconds).

**2. Setting Up the Environment:**

* **Language Choice**: C was chosen for this project due to its efficiency, familiarity, and control over low-level operations (like memory management). It’s also well-suited for console-based applications.
* **Libraries**: The program uses standard C libraries for input/output (stdio.h), memory allocation (stdlib.h), string manipulation (string.h), character handling (ctype.h), and time (time.h). Windows-specific libraries (windows.h) are used for timer management and console color manipulation.

6

**3. Reading and Storing Questions:**

* **File Handling**: The questions are read from a text file (questions.txt). The file is parsed line by line to extract:
  + Question text.
  + Answer options.
  + The correct option.
  + Timeout value.
* This information is stored in an array of Question structs.
* **Dynamic Memory Allocation**: The Question array is dynamically allocated to store up to TOTAL\_QUESTIONS (300). This allows the program to scale if more questions are added to the file.

**4. Game Logic Implementation:**

* **Shuffling Questions**: To make the game less predictable, the questions are shuffled randomly before the game begins. This ensures that the order of the questions varies with each game.
* **Timer Setup**: A countdown timer is set for each question using Windows-specific CreateTimerQueue and CreateTimerQueueTimer. The timer runs asynchronously, and if the time expires, it triggers the timeout\_handler callback to end the game.
* **Answer Validation**: After the timer starts, the player is prompted to answer. The answer is checked against the correct option:
  + If the player answers correctly, they earn the prize money associated with the question.
  + If the answer is incorrect, the game ends immediately.
  + The game also handles player lifelines (50/50 and Skip) to help them during the game.

**5. Lifeline Implementation:**

* **50/50 Lifeline**: When the player selects the 50/50 lifeline, two incorrect options are removed randomly, leaving only the correct answer and one wrong answer.
* **Skip Lifeline**: If the player opts for the Skip lifeline, they can skip the current question and proceed to the next one without answering.

7

* These lifelines are tracked with an array (lifeline[]), where each lifeline is represented by a boolean value (1 for available, 0 for used).

**6. Player Interaction and UI:**

* **Console Output**: The game outputs text to the console in a visually appealing way, using colored text and styled lines to make the game feel more engaging. The SetConsoleTextAttribute function is used to change text color based on the status of the game (e.g., green for correct answers, red for wrong answers).
* **Input Handling**: The program uses scanf to take the player's input (answer or lifeline choice). It ensures that the player's input is validated and correctly processed.

**7. End of Game Handling:**

* When the game ends, either because the player answered incorrectly or ran out of time, a summary of the player's total winnings is displayed.
* The results are saved to a file (results.txt) to maintain a log of players’ scores.

**8. File Management:**

* **Questions File**: The program expects a file (questions.txt) containing all the questions. The file format is structured to include the question, four answer options, the correct answer, and the timeout value for each question.
* **Results File**: After the game ends, the player’s name and total winnings are appended to a file (results.txt), storing a record of the player’s performance.

**9. Testing and Debugging:**

* **Basic Functionality Testing**: The first phase of testing ensures that the game works as expected, including reading questions, displaying them, accepting user input, handling the timer, and displaying results.
* **Edge Case Testing**: Testing involves scenarios like:
  + What happens if the player doesn’t answer in time.
  + What happens if the player selects an incorrect answer.
  + Checking that the timer countdown works as expected.
  + Ensuring that lifelines work properly (50/50 and Skip).
* **Memory Management**: Ensuring that dynamic memory is allocated and freed properly to avoid memory leaks.

8

**10. Improving the User Experience:**

* **Text Styling**: The game uses color and formatting to make the console output more engaging. The use of printf with color codes and styled lines helps create a more visually appealing interface.
* **Player Feedback**: Providing clear feedback to the player on their performance (correct/incorrect answers, prize money won, etc.) enhances the experience

The development process involves multiple steps, including designing the game flow, implementing the core logic (question handling, timers, lifelines), testing the functionality, and improving user experience. Each step is carefully planned and executed, ensuring the game is engaging.

9

**PROJECT SCOPE**

The **scope of the quiz game project** outlines its boundaries, objectives, and deliverables. It defines what the project will and will not cover, helping stakeholders understand the project's direction, time frame, and key features. Here’s an overview of the project scope.

**1. Objective of the Project**

* **Main Goal**: To develop an interactive, engaging quiz game that challenges players’ knowledge across various topics, providing them with an enjoyable experience and potential rewards (money).
* **Target Audience**: General public, players looking for entertainment, students, and individuals interested in testing their general knowledge.

**2. Key Features**

The quiz game project will include the following core features:

* **User Interface**:
  + A text-based user interface (for the initial version).
  + Display of questions, options, and the countdown timer.
  + Real-time feedback on player’s answer (correct/incorrect).
  + Lifeline options (50/50, Skip) for user assistance.
  + Display of current money earned after each questions.

**3. Exclusions**

* **Advanced Features**:
  + Multiplayer mode.
  + Graphical User Interface (GUI) beyond the command-line interface.
  + Mobile or web versions of the game.
  + Support for multiple languages.
* As part of future iterations, additional features can be added as mentioned before, such as multiplayer modes, a mobile app version, GUI enhancements, and more advanced question databases. However, these improvements are outside the scope of the current project phase.

**PROJECT SCHEDULE**

**Timeline**

* Week 1:
* Requirement analysis and design.
* Development of question.txt file.
* Initial implementation of user interaction logic
* Week 2
* Completion of user interaction logic.
* Testing and bug fixing.

10