**Project Name : " QUIZ MASTER "**

**Team member ID :**

**1. 242-35-555**
**2. 242-35-525**
**3. 242-35-167**

**Project overview**

- **Objective :** The Quiz Game is an interactive C-based console application designed to test a user's general knowledge through a series of multiple-choice questions. The game provides a straightforward, user-friendly experience where the player can select a difficulty level, answer questions, and track their score. It is ideal for anyone looking to test their knowledge on a variety of topics while offering a fun and educational experience.

- **Features**
  * User selects difficulty (Easy, Medium, or Hard).
  * Questions are displayed with options A, B, C, D.
  * User inputs an answer, and the program checks if it's correct.
  * Score is updated and displayed after each question.
  * After all questions, the final score and percentage are displayed.
  * The player is asked if they want to play again. If yes, the quiz restarts.

  **Structure and Functionality**
  The code is structured into different parts:

  **Input**

  - The user provides input for:
    1. Selecting the difficulty level (`scanf("%d", &choice);`).
    2. Answering quiz questions (`scanf(" %c", &answer);`).
    3. Deciding whether to replay (`scanf(" %c", &playAgain);`).

  **Why is this needed?**
- User input is essential to make the program interactive. It allows dynamic selection of difficulty levels and lets users participate actively.

**Logic**

1. Struct for Questions – The **struct Question** is used to store questions, multiple-choice options, and the correct answer.

2. Functionality Flow:

    - The quiz game starts and asks the user to choose a difficulty level.
    - The appropriate set of questions is selected and stored in an array.
    - Each question is displayed one by one, and the user's input is taken.
    - If the answer is correct, the score is incremented.
    - The final score and percentage are displayed after all questions.
    - The user is given the choice to play again.

**Why is this needed?**

- The logic ensures a structured and user-friendly experience. It follows a logical order to handle inputs, process the quiz, and display outputs efficiently.

**Output**

- Displays quiz questions with multiple-choice options.
- Shows the user's score after each question.
- At the end, it displays:

    1. The final score.
    2. The percentage.
    3. A prompt asking if the user wants to replay.

**Why is this needed?**

- Feedback is crucial for user engagement. Displaying scores and percentages helps users track their performance.

## Identifying Key Elements

### Operators Used

The code uses various operators:

- Assignment Operator ( `=` ) – Used to assign values (e.g., `quiz[0].correctAnswer = 'A';` ).
- Arithmetic Operators ( `+` , `/` , `*` ) – Used in score calculation and percentage ( `float percentage = (float)score / totalQuestions * 100;` ).
- Comparison Operator ( `==` ) – Used to check if the user's answer is correct ( `if (answer == q.correctAnswer)` ).
- Logical Operator ( `&&` , `||` ) – Not used in this version, but could be useful for further validation.

Why are they used?

- These operators help in assigning values, performing calculations, and making decisions based on user inputs.

## Conditional Statements

1. `if` Statements
   - Used to check if the answer is correct:

```
if (answer == q.correctAnswer) {
    correct = 1;
}
```

Used in replay functionality:

```
if (playAgain == 'Y')
```

2. `switch` Statement

- Used to select the difficulty level and load the appropriate questions.

```
switch (choice) {
    case 1:
        // Load easy questions
        break;
    case 2:
        // Load medium questions
        break;
    case 3:
        // Load hard questions
        break;
}
```

**Why are they used?**

- if statements are necessary for decision-making at multiple points.
- The switch statement simplifies handling multiple difficulty levels efficiently.

**Loops**

`for` Loop

- Used to iterate through quiz questions:

```
for (i = 0; i < totalQuestions; i++) {
    correct = askQuestion(quiz[i]);
    score += correct;
}
```

`do-while` Loop

- Used to keep playing until the user decides to stop:

```
do {
    // Run the quiz
} while (playAgain == 'Y');
```

**Why are they used?**

- The for loop ensures that all questions are asked in sequence.
- The do-while loop ensures that the game runs at least once and allows replaying.

**Arrays and Strings**

- **Arrays**
  - The quiz questions and options are stored in an array:

```
struct Question quiz[3];
```

- **Strings**
  - Used to store question text and answer choices (`char question[200];`).
  - String functions like `strcpy()` are used to assign values.

**Why are they used?**

- Arrays store multiple questions efficiently.
- Strings are needed for handling text-based input/output.

## Explanation of the Code

1. **Defining the `Question` Structure**
   - Holds the question, four answer choices, and the correct answer.
2. **Function `displayScore()`**
   - Calculates and prints the final score and percentage.
3. **Function `askQuestion()`**
   - Displays the question and options, takes user input, and checks correctness.
4. **Function `selectDifficulty()`**
   - Lets the user choose a difficulty level and loads corresponding questions.
5. **Main Function (`main()`)**
   - Controls the entire game flow:
     - Calls `selectDifficulty()`.
     - Uses a loop to ask questions and track the score.
     - Displays the final score.
     - Allows the user to replay.

## Summary of Why Each Component is Used

| Component | Purpose |
|---|---|
| `struct Question` | Stores quiz details. |
| `displayScore()` | Displays results. |
| `askQuestion()` | Handles user input and answer checking. |
| `selectDifficulty()` | Loads questions based on user choice. |
| `main()` | Runs the quiz and handles replaying. |
| `for` loop | Iterates through questions. |
| `do-while` loop | Allows replaying the game. |
| `if` statements | Checks user answers and game continuation. |
| `switch` statement | Selects difficulty level efficiently. |
| `Arrays` | Stores multiple questions. |
| `Strings` | Handles text for questions and options. |

**CODE :**

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
struct Question {
    char question[200];
    char options[4][100];
    char correctAnswer;
};

void displayScore(int score, int totalQuestions) {
    float percentage = (float)score / totalQuestions * 100;
    printf("\nYour total score: %d out of %d\n", score, totalQuestions);
    printf("Your percentage: %.2f%%\n", percentage);
}

int askQuestion(struct Question q) {
    char answer;
    int correct = 0;

    printf("\n%s\n", q.question);
    printf("A. %s\n", q.options[0]);
    printf("B. %s\n", q.options[1]);
    printf("C. %s\n", q.options[2]);
    printf("D. %s\n", q.options[3]);
    printf("Your answer (A, B, C, D): ");
    scanf(" %c", &answer);
```

```
    answer = toupper(answer);
    if (answer == q.correctAnswer) {
        correct = 1;
    }

    return correct;
}
void selectDifficulty(struct Question quiz[]) {
    int choice;
    printf("Welcome to the Quiz Game!\n");
    printf("Select your difficulty level:\n");
    printf("1. Easy\n");
    printf("2. Medium\n");
    printf("3. Hard\n");
    printf("Enter your choice (1/2/3): ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
        // Easy questions
            strcpy(quiz[0].question, "What is the capital of France?");
            strcpy(quiz[0].options[0], "Paris");
            strcpy(quiz[0].options[1], "London");
            strcpy(quiz[0].options[2], "Berlin");
            strcpy(quiz[0].options[3], "Madrid");
            quiz[0].correctAnswer = 'A';
```

```
    strcpy(quiz[1].question, "Which planet is known as the Red Planet?");
    strcpy(quiz[1].options[0], "Earth");
    strcpy(quiz[1].options[1], "Mars");
    strcpy(quiz[1].options[2], "Venus");
    strcpy(quiz[1].options[3], "Jupiter");
    quiz[1].correctAnswer = 'B';

    strcpy(quiz[2].question, "What is 2 + 2?");
    strcpy(quiz[2].options[0], "3");
    strcpy(quiz[2].options[1], "4");
    strcpy(quiz[2].options[2], "5");
    strcpy(quiz[2].options[3], "6");
    quiz[2].correctAnswer = 'B';
    break;

case 2:
    strcpy(quiz[0].question, "Who wrote 'To Kill a Mockingbird'?");
    strcpy(quiz[0].options[0], "Harper Lee");
    strcpy(quiz[0].options[1], "J.K. Rowling");
    strcpy(quiz[0].options[2], "Mark Twain");
    strcpy(quiz[0].options[3], "F. Scott Fitzgerald");
    quiz[0].correctAnswer = 'A';

    strcpy(quiz[1].question, "What is the chemical symbol for water?");
    strcpy(quiz[1].options[0], "H2O");
    strcpy(quiz[1].options[1], "CO2");
    strcpy(quiz[1].options[2], "O2");
```

```
    strcpy(quiz[1].options[3], "NaCl");
    quiz[1].correctAnswer = 'A';

    strcpy(quiz[2].question, "Which element has the atomic number 1?");
    strcpy(quiz[2].options[0], "Helium");
    strcpy(quiz[2].options[1], "Hydrogen");
    strcpy(quiz[2].options[2], "Oxygen");
    strcpy(quiz[2].options[3], "Carbon");
    quiz[2].correctAnswer = 'B';
    break;

case 3:
    strcpy(quiz[0].question, "What is the fastest land animal?");
    strcpy(quiz[0].options[0], "Lion");
    strcpy(quiz[0].options[1], "Cheetah");
    strcpy(quiz[0].options[2], "Leopard");
    strcpy(quiz[0].options[3], "Elephant");
    quiz[0].correctAnswer = 'B';

    strcpy(quiz[1].question, "Who developed the theory of relativity?");
    strcpy(quiz[1].options[0], "Isaac Newton");
    strcpy(quiz[1].options[1], "Albert Einstein");
    strcpy(quiz[1].options[2], "Galileo");
    strcpy(quiz[1].options[3], "Nikola Tesla");
    quiz[1].correctAnswer = 'B';

    strcpy(quiz[2].question, "Which programming language is known as 'C's successor'?");
```

```
            strcpy(quiz[2].options[0], "Java");
            strcpy(quiz[2].options[1], "C++");
            strcpy(quiz[2].options[2], "Python");
            strcpy(quiz[2].options[3], "Swift");
            quiz[2].correctAnswer = 'B';
            break;
    }
}
int main() {
    struct Question quiz[3];
    int score = 0, totalQuestions = 3, i, correct;
    char playAgain;

    do {
        selectDifficulty(quiz);
        printf("Quiz Started! Good luck!\n");

        for (i = 0; i < totalQuestions; i++) {
            correct = askQuestion(quiz[i]);
            score += correct;
            printf("Your score so far: %d\n", score);
        }
        displayScore(score, totalQuestions);
        printf("\nDo you want to play again? (Y/N): ");
        scanf(" %c", &playAgain);
        playAgain = toupper(playAgain);
       } while (playAgain == 'Y');
```

```
    printf("Thanks for playing! Goodbye!\n");
    return 0;
}
```

## CONCLUTION

The **Quiz Game** is an interactive and user-friendly program that tests a player's knowledge across different difficulty levels. It provides a structured gameplay experience by incorporating **question selection, user input handling, answer validation, and score tracking**. The game ensures a smooth flow by using structured programming concepts such as **loops, conditional statements, functions, and arrays**.

Key strengths of the game include:

1. **Dynamic Difficulty Selection** – Allows players to choose between Easy, Medium, and Hard levels.
2. **Interactive Question-Answer Format** – Engages users with multiple-choice questions.
3. **Real-Time Score Updates** – Keeps players informed of their progress.
4. **Replay Feature** – Encourages repeated play for learning and improvement.

Overall, this quiz game serves as a great example of fundamental programming concepts in **C** while also being a fun and engaging way to test knowledge. It can be further expanded by adding **a larger question bank, a timer for answering questions, and a leaderboard system** to enhance the experience.