**PROJECT REPORT**

**Topic:**

**Quiz System**

**Overview**

The Quiz System program is designed to create a quiz environment where users can answer multiple-choice questions. The program presents a series of questions with options, and the user selects the correct answer by choosing the corresponding option. The program also includes features such as user login and password validation.

The program starts by displaying a welcome message and prompting the user to enter their name, login ID (Roll\_no), and a 4-digit password. Once the user enters the required information, they are asked to press any key to continue.

After the initial setup, the program displays the main menu, which offers three levels of

corresponding option. Based on the user's selection, the program branches to the respective quiz level.

The Easy, Medium, and Hard quiz levels consist of a set of multiple-choice questions. Each question is presented along with four options, and the user is required to choose the correct option. The program validates the user's answer and provides feedback regarding the correctness of their response.

At the end of the quiz, the program displays the user's score out of 10. The score indicates the number of correctly answered questions. A message of "Good" is shown to encourage the user based on their performance.

The program includes additional information such as the names of the developers who designed it. The design of the program follows a structured approach, with clear separation of data and code sections.

Overall, the Quiz System program provides an interactive and educational quiz experience. It tests the user's knowledge on various topics and provides immediate feedback on their performance. The program's user-friendly interface and well-designed structure contribute to an engaging and enjoyable quiz experience.

DETAILED IMPLIMENTATION

**.Model small:** This line specifies the memory model used for the program.

**.Stack 100h**: This line sets the size of the program stack to 100 hexadecimal bytes.

**.data**: This line marks the beginning of the data section, where variables and strings are defined.

**NameStr db** " Enter your name: $",0: This line defines a variable called "NameStr" and assigns it the string value "Enter your name: " followed by a dollar sign and null character.

**LoginId db** " Enter your login id (Roll\_no): $",0: This line defines a variable called "LoginId" and assigns it the string value "Enter your login id (Roll\_no): " followed by a dollar sign and null character.

**Password db** " Enter password (4-digit code): $",0: This line defines a variable called "Password" and assigns it the string value "Enter password (4-digit code): " followed by a dollar sign and null character.

**Continue db** " Press any key to continue.... $",0: This line defines a variable called "Continue" and assigns it the string value "Press any key to continue.... " followed by a dollar sign and null character.

**Exit db** " Press any key to exit..... $",0: This line defines a variable called "Exit" and assigns it the string value "Press any key to exit..... " followed by a dollar sign and null character.

**string\_00 db** " Hi Welcome! $",0: This line defines a variable called "string\_00" and assigns it the string value "Hi Welcome! " followed by a dollar sign and null character.

**Main Proc**

This line defines the start of a procedure named "Main."

mov ax,@data mov ds,ax

These two lines are used to set up the data segment by moving the value of the @data symbol into the ax register and then transferring it to the ds register.

**MOV AX, 07h MOV BH, 07h MOV CX, 0 MOV DX, 184Fh INT 10h**

These instructions are responsible for clearing the screen. The MOV instructions set the values of the AX, BH, CX, and DX registers, and then the INT 10h instruction triggers an interrupt that clears the screen.

**call LineBreak**

These two lines are function calls to a procedure named LineBreak, which is not shown in the provided code. It is likely that LineBreak inserts line breaks or newlines in the output.

**Mov si,0 mov dx,offset string\_00 mov ah,9h int 21h**

These lines initialize the si register to zero, load the address of the string string\_00 into the dx register, set the value 9h (indicating a print string function) in the ah register, and trigger an interrupt int 21h to print the string. The subsequent call LineBreak statements insert line breaks.

**Mov ah,9 mov dx,offset NameStr int 21h mov ah,0Ah int 21h**

These lines print a string stored at the memory address NameStr. The mov instructions set the necessary values for printing a string, and the int 21h interrupt is called to display the string. The mov ah,0Ah and the second int 21h are used to input a string from the user, storing it at the memory address specified by NameStr. The subsequent call LineBreak statements insert line breaks.

**Mov ah,09h mov dx,offset loginId int 21h mov ah,0AH int 21h**

These lines are similar to the previous section but are used to input and display the login ID. The string is stored at the memory address specified by loginId.

**Mov ah,09h mov dx,offset password int 21h mov cx,4 Label\_: mov ah,1 int 21h loop label\_**

These lines display and input the password. The mov instructions set up the required values, and the interrupt int 21h is called to display the string stored at the memory address password. The next instructions set up a loop that reads a character input (password) using interrupt int 21h with function ah=1 and repeats it four times using the loop instruction.

**Mov ah,09 mov dx,offset continue int 21h mov ah,08h int 21h**

These lines display a string stored at the memory address continue, prompting the user to continue. The user's response is then read using interrupt int 21h with

* **Easy Quiz**

**EasyQuizProc Proc**

This line indicates the start of a procedure called "EasyQuizProc," which is related to an easy quiz.

call LineBreak call LineBreak

These two lines are function calls to a procedure or function named "LineBreak" to insert line breaks or newlines in the output.

**Mov dx,offset string\_05 mov ah,9h int 21h**

These lines load the memory address of the string "string\_05" into the dx register, set the value 9h (indicating a print string function) in the ah register, and trigger an interrupt int 21h to print the string.

**Mov dx,offset Q\_1 mov ah,9h int 21h**

These lines display a question (stored at the memory address Q\_1) by loading its address into the dx register, setting the value 9h in the ah register, and triggering an interrupt int 21h to print the string.

**Mov dx,offset option1 mov ah,9h int 21h**

These lines display an option (stored at the memory address option1) by loading its address into the dx register, setting the value 9h in the ah register, and triggering an interrupt int 21h to print the string.

**jmp input1**

This line jumps to a label named "input1" (presumably for receiving user input related to the first question).

Input1: mov dx,offset string\_06 ; for choose mov ah,9h int 21h mov ah,1 int 21h mov bl,0 mov bl,al cmp bl,'a'

**jz TRUE1 jmp False1**

These lines display a prompt (stored at the memory address string\_06) to choose an option. The user's input is then read using interrupt int 21h with function ah=1. The value is stored in the al register and then moved to the bl register. The subsequent lines compare the value in bl with the character 'a' and jump to either **"TRUE1" or "False1**" based on the comparison result.

The code following "False1" and "TRUE1" branches displays appropriate messages related to the user's choice and the correct answer.

The subsequent sections of the code follow a similar pattern, displaying questions, options, receiving user input, and providing feedback based on the user's choices.

The labels "L2" and "L3" are used to jump to the next set of questions and options.

**Jmp input3**

This line jumps to a label named "input3," suggesting that it is related to receiving user input for the third question.

**Uptill L10" is reached.**

The process repeats for the next questions and user inputs until "L10" is reached.

* **Medium & Hard Quiz**

Same procedure is followed for Medium and Hard Quiz

**Score Calculating**

1. The code begins at the label "L11," assuming the program execution has reached this point from a previous section.
2. The code calls the "LineBreak" subroutine twice, which likely inserts two line breaks or newline characters for formatting.
3. The next section of code displays a line of characters using the DOS interrupt 21h function 9h. The message is stored in the "lines1" variable, which is assumed to be a placeholder for the line.
4. After displaying the line, the code calls "LineBreak" again for formatting and displays a score message using a similar approach. The message is stored in the "string13" variable.
5. The code moves the value of the "si" register (possibly representing the score) to the "totalscore" variable.
6. The code compares the value in "totalscore" with the value 10 using the "cmp totalscore, 10" instruction.
7. If the comparison results in equality (score equals 10), the code jumps to the "Full" label. Otherwise, it proceeds to display the score.
8. If the score is less than 10, the code converts the score to ASCII representation by adding 30h (hexadecimal value of '0') to each digit and displays it using DOS interrupt 21h function 2h. It then displays the message stored in the "string14" variable using a similar approach.
9. The code calls "LineBreak" for formatting, displays the line stored in the "lines1" variable, and prompts the user to continue using the message stored in the "continue" variable.
10. The code waits for user input using DOS interrupt 21h function 8h and clears the screen using the "INT 10h" interrupt with appropriate register values.
11. It calls the "LineBreak" subroutine for formatting and proceeds to display the designed message, which is stored in multiple string variables (string16, string17, string18, string19), using the DOS interrupt 21h function 9h.
12. After displaying the design messages, the code calls "LineBreak" twice for additional formatting.
13. It displays a closing message stored in the "string12" variable using DOS interrupt 21h function 9h.
14. The code calls "LineBreak" twice for formatting.
15. It prompts the user to continue using the message stored in the "continue" variable and waits for user input using DOS interrupt 21h function 8h.
16. The code clears the screen by invoking the "INT 10h" interrupt with appropriate register values.
17. It calls the "LineBreak" subroutine for formatting.
18. The code displays the designed messages again, similar to the previous section.
19. It calls "LineBreak" twice for formatting.
20. Finally, the code displays a closing message stored in the "string12" variable using DOS interrupt 21h function 9h.