# Quiz

## Task Brief

A primary school teacher wants a computer program to test the basic arithmetic skills of her students. The program should generate a quiz consisting of a series of random questions, using in each case any two numbers and addition, subtraction and multiplication. The system should ask the student’s name, then ask 10 questions, output if the answer to each question is correct or not and produce a final score out of 10.

## Introduction

You can draw these conclusions from the brief,

* You are making a quiz program for primary school pupils. The questions do not involve division and, because the pupils are young, you can expect the answers to the questions to be positive, whole numbers, the integer data type. Due to the age of the pupils, you should base your multiplication questions on times tables and only use numbers between 1 and 12. For the addition and subtraction, you can use a bigger range of numbers - you will need to make a choice.
* For subtraction, the larger number must go first so that the answer is positive.
* You will need to be able to write programs that generate random numbers. You will have to choose two random numbers for each question and choose which arithmetic operation to use.
* You have to choose the operation before picking the numbers for the question. This is because there is a different range of numbers for multiplication.
* After you make each question, you will have to ask it to the user and read their answer. You will have to check if their answer is correct and add 1 to the score if it is.
* There are 10 questions. You will have to repeat the process of making, asking and checking a question exactly 10 times. You will need to use a FOR loop to repeat the statements that do this.
* At the end of the quiz, the pupil has to be told their score out of 10.

Explain this information in your own words, using the heading **Introduction**. You need to spend **no more than 15 minutes** doing this. Type quickly and make any corrections you need at the end rather than whilst you are typing.

## List Of Steps

Here is a list of steps to help you understand how this program will have to work. You can copy and include this in your work under the heading **List Of Steps**. It is a very rough summary of what the program will be like.

*Ask the user their name*

*Input the user's name*

*Set the score to 0*

*For each question (1 - 10)*

*Choose a random operation*

*If the operation is 0 (multiply), choose numbers between 1 and 12. Choose a bigger range of numbers if not.*

*Work out the answer.*

*Ask the question.*

*Input the user's answer.*

*Check the answer and add 1 to the score if it is correct.*

*End the loop*

*Output user's name and score out of 10.*

## Variable List

Copy the following table into your work underneath a heading **Variable List**. Fill in the second column with the correct data types. You will need to write **String** for anything that stores letters and **integer** for anything that stores numbers. For the last column, finish the sentences that begin with the word **Stores** to say what information the variable should store. This should take no more than 10 minutes.

|  |  |  |
| --- | --- | --- |
| ***Variable Name*** | ***Data Type*** | ***Purpose*** |
| *operation* |  | Stores |
| *a* |  | Stores |
| *b* |  | Stores |
| *largest* |  | Stores |
| *smallest* |  | Stores |
| *score* |  | Stores |
| *answer* |  | Stores |
| *useranswer* |  | Stores |
| *username* |  | Stores |
| *question* |  | Loop counter variable. |

## Development Stage 1 - Generating The Numbers For A Question

In Visual Basic, we can generate pseudo-random numbers using the Random class. To start, declare a new instance of the Random class in the following way,

*Dim rand As Random = New Random()*

You only need this line once in the program, best done at the start.

To generate a random number, set the variable that stores it, like this,

*myRandomNumber = rand.Next(1,13)*

This makes a random number from 1 to 12. The second number has to be one more than the highest number you want to make.

1. Write enough code to generate and output 2 random numbers in the range 1-12, storing the values in variables **a** and **b**.
2. Add more statements to generate 2 more numbers in the range 1-50. Use the same a and b variables.
3. Test your program. You MUST make sure that you generate the correct range of numbers. It will only be by not reading a single sentence that you could get this wrong. Show how you tested the program.

## Development Stage 2 – Making A Proper Question

This is quite a large stage. Your program will,

Choose an operation (add, subtract, multiply) at random and make a suitable question.

You can begin this stage by declaring all of the variables listed previously with the exception of the variable called **question**.

Make all of the number variables equal to 0. Make the string variable equal to “” (empty quotation marks).

You will need to remove/move the lines of code you wrote at the previous stage.

Start by making the variable **operation** equal to a random number from 0 to 2. You MUST take the trouble to ensure that you generate the correct range of numbers. It only requires you to read the single sentence of explanation written earlier. There are no excuses for getting this wrong.

Here is the Scratch code for making an addition question,



Notice that we have to work out the answer to the question here.

You also need to display the question.

We are not getting any user input at this stage.

Now for multiplication,



When it comes to multiplication, the numbers we use are different.

Finally, subtraction, the one that is a little more complicated,



When we subtract, the question uses the largest and smallest variables instead of a and b.

That’s how we ensure that the questions do not result in a negative number.

1. Get all of this code converted to Visual Basic. Test your program and show that it can produce suitable questions.

## Development Stage 3 – The User’s Answer

1. The code so far, should generate and ask a question. After the last of the IF statements is concluded, you need to read in the user’s answer. This happens once. Although there are 3 types of question, only one of them gets asked, so we only need to read in the answer once. The variable you use to store the user’s answer is called **useranswer**.
2. Next you need to check if the user’s answer matches the genuine answer, stored in the variable **answer**. If it does, tell the user they are correct and increase the score by 1. If the user is wrong, tell them they are incorrect.
3. Test the program at this stage.

## Development Stage 4 – 10 Questions

The quiz is supposed to consist of 10 questions.

1. The code that generates a single question for the quiz begins with a random choice of operation. It ends after the user has been told whether they are right or wrong. This code needs to be enclosed within a For..NEXT loop. Use the variable name **question** for the loop counter and make it count from 1 to 10.
2. Test that your program produces 10 questions, that the questions are different and that the answers are correctly judged.

## Development Stage 5 – User’s Name & Score

1. Before the quiz begins, get the user to enter their name. At the end of the quiz, output the user’s name and score for the test.
2. Test your program thoroughly and explain how you know that it works correctly.

## Extension 1 – Division Questions

The quiz does not include any division questions. This to ensure that the primary school pupils don’t have to work out anything with decimals.

If you can generate division questions that do not include decimals, you could include them in the quiz.

Think about the multiplication questions. You made sure that the questions you asked were all part of the multiplication tables. The two numbers you choose for **a** and **b** are factors of the answer – they have to be. So here is an idea,

*answer = random number from 1 to 12*

*b = random number from 1 to 12*

*a = answer \* b*

1. Use this information to extend the program to include division questions.

## Extension 2 – Another Go

1. Change the program so that the player can choose to play again without having to restart the program.

## Extension 3 – Timed Quiz

Instead of having a specific number of questions, you can make the quiz so that you get a time limit.

The stopwatch component can be used for that. The following example will make the computer count from 1 upwards for exactly 5 seconds.

 Dim n As Long

 Dim s As Stopwatch = New Stopwatch()

 n = 1

 s.Reset()

 s.Start()

 Do

 Console.WriteLine(n)

 n = n + 1

 Loop Until s.ElapsedMilliseconds >= 5000

 s.Stop()

 Console.WriteLine("Done")

 Console.ReadLine()

1. Copy this code into a **new** solution and run it several times. Why do you think that it stops at a slightly different number each time?
2. Use this example to help you make a **new** quiz program that gives the player 2 minutes to answer as many questions as possible.