# Lessons 5 and 6 – Activity Sheet

Setting the Scene

In this project we will make a program that gives **multiplication** maths problems and answers to help young Learners practice their times tables.

## Success Criteria

* Has a simple interface when the program starts
* Pressing a button gives a random multiplication problem using numbers between 1 and 10
* Pressing another button gives the answer

Open <https://makecode.microbit.org/>

The first thing to consider here is how the program will be used. The success criteria requires a simple interface, so we need to have some very simple instructions when the program first runs. As the A and B buttons are either side of the LEDs we will program Button A to give the question (Q) and Button B give the answer (A) and show this with a simple arrow:

Graphical user interface, text, application, chat or text message

Description automatically generated

We now need to program Button A. The success criteria requires the numbers be generated randomly so we will need to use a variable to temporarily store this value.

## Variables

A variable is a container for a value, like a number which we could use in a sum, or a **string** (text) that we might use as part of a sentence. One special thing about variables is that their contained values can change.

We need to create two variables to store the two **random** numbers needed for a multiplication problem:

A close up of a logo

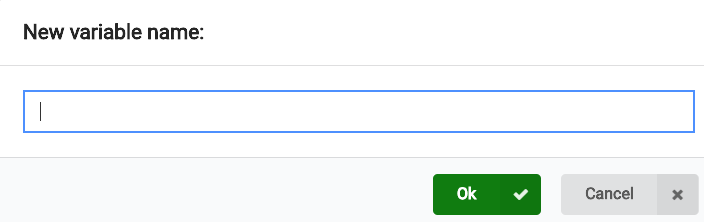
Description automatically generatedTo create a variable you need to open the **Variables** menu

in MakeCode and select **Make a Variable**

A close up of a sign

Description automatically generated

this will open a menu to name the variable:



Pro-tip

Name your variables wisely! A good variable name describes what it is for or what it contains. Variable names should be lowercase, with words separated by underscores to improve readability.

***my\_first\_variable*** is better than ***My1stVar***

In this example we have named the variables ***int1*** and ***int2.*** Int is short for **integer** which is a computing/maths word for whole number.

|  |  |
| --- | --- |
| Once you have created the two variables they will appear in the **Variables** menu to be used with other blocks. | A screenshot of a cell phone  Description automatically generated |

Now you need to use an ***on button A pressed*** block and inside that put a ***set to*** block for both ***int1*** and ***int2*** and then attach a ***pick random number*** from the **Maths** blocks. Next change the number from 4 to 10 in the ***pick random*** block.

A screenshot of a cell phone

Description automatically generated

This will set the two variables to random numbers from 1 to 10 when button A is pressed. We now need to show this on the LEDs so the user can see the question.

|  |  |
| --- | --- |
| A picture containing monitor, remote  Description automatically generated | Now we can add the ***show number*** blocks to show the random numbers.  To make the program suitable for younger learners we have added a ***pause*** block so the number stays on the LEDs long enough to read.  The ***show string “x”*** block represents the multiplication symbol.  That last block is another ***show string*** that displays the “=” sign and then a “?” to show that it is a question. |

The process so far:

* User is prompted to press button A for a question

We now need to display the answer as this is also required by the success criteria. We will now program button B to give the answer. As we have stored the randomly generated numbers in variables when button A is pressed we can now multiply them together and show this on the LEDs.

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated | Again we use an ***on button pressed*** block but this time for button B and we also add the ***show number*** block. Inside that block we use another **Maths** block to multiply ***int1*** with ***int2***. |

## Test Time

We have used a few different blocks here so now is the time to test your program and make sure that it behaves as you would expect.

Give your program a name such as **math\_app.hex** and **Download** it onto you micro:bit.

Make sure to check that the answer is correct a few times! If not, look at the blocks and see where the problem is and fix it as you go along.

## Stretch Tasks

* Make the program use the other maths operators (+, -, /) you could make the program change the operator when shaken for example
* Make the micro:bit play a sound prior to the answer being displayed
* Make another microbit keep score for two players (you will need to use the **Radio** blocks)
* Add a hard mode (questions between 0 and 100) when A+B are pressed

## Final Thoughts

In this project we have covered:

* Inputs
* Variables
* Random numbers
* Basic maths
* Testing and troubleshooting

And combined it to make a useful app. This is what computing is all about, using the tools to make something useful.