Using the Arduino for Math

Before you get started, [read the relevant sections of this document](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit?usp=sharing), to support your understanding.

Lots of quick experiments can be done with this simple Arduino sketch that will have you get the basics of how the programming works. As evidence for these experiments, you will need to collect a series of screenshots to show that you have completed them. You will also need to get an actual photograph of how you’ve connected the Arduino to the computer with a USB cable. The laptop’s webcam or a phone can get the Arduino’s photo, but screenshots should be take with the operating system’s utility to do so. On windows, the easiest option is to use the “Snipping Tool.” Save all these screenshots, so you can submit them to show your work.

# Example Sketch 1: Addition on Microcontroller

Below is an example some simple addition on the Arduino. You can copy/ paste this into your Arduino IDE

/\*

\* A basic math example, that shows the need for variable types

\*/

**int a = 35;** //global variable called ‘a’ to store first input

**int b = 4;** //global variable called ‘b’ to store second input

**void setup(){** //this will run only once

**Serial.begin(9600);** //start and set-up serial communication

**int c;** //declare a variable called ‘c’ without a value

**c = a + b;** //this is where the math happens << edit this line

**Serial.println(c);** //return the output over serial

**}**

**void loop(){** //you can totally ignore this, but it must be here

**}**

The program doesn’t do too much. Just a little math, and a serial monitor response. This version just adds the two numbers 35 and 4 and then sends the result back to the serial monitor. This program can be edited and changed in many ways. Changing the orange line of code actually alters the math that is done. In order to finish the assignments, you will also need to change the green variable types, for a few of the assignments.

Learning how to find the important lines of code and ‘tune out’ the rest is an important skill while you get started. The highlighted code in the example above is meant to help with that...

# Many Tasks to Complete

## Perform addition on the microcontroller

Type the program above so that the Arduino does some simple addition. Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot.

Screenshot goes here

Which line of code actually does the addition in the program?

Copy the line of code that does the addition here

## Modified addition on the microcontroller

Type the program above so that the Arduino does some simple addition but change out the numbers. Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot.

Screenshot goes here

Which line of code actually sends a message back thought the Serial port in the program?

Copy the line of code that does the addition here

## ‘Over the memory limit’ addition on the microcontroller

Type the program above so that the Arduino does some simple addition, but make sure that the sum of the two numbers exceeds the [maximum positive value of an int](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.ry2zxjrodesn). Just to see what happens as a result.

An example sketch of this is included below, but your screenshot should show that you at least changed out these numbers, while still having ‘over the memory limit’ addition. 32,767

**int a = 32,765;** //global variable called ‘a’ to store first input

**int b = 5;** //global variable called ‘b’ to store second input

**void setup(){** //this will run only once

**Serial.begin(9600);** //start and set-up serial communication

**int c;** //declare a variable called ‘c’ without a value

**c = a + b;** //this is where the math happens << edit this line

**Serial.println(c);** //return the output over serial

**}**

**void loop(){** //you can totally ignore this, but it must be here

**}**

Take a screenshot of this ‘over the memory limit’ sketch working with your own numbers, and make sure that the program and serial monitor results are visible in the screenshot.

Screenshot goes here

Citing textual evidence from the article linked in the prompt above, why does the this example of addition not work as you would expect it to in a typical math-class calculator?

Explain the reason for the ‘over the memory limit’ behavior you observed in the screenshot here

## Longer addition on the microcontroller

Type the program above so that the Arduino does some simple addition, but [make the variables long, unsigned, or both](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.ry2zxjrodesn) to see how this changes the operation. Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot.

Screenshot goes here

Cite text evidence from the link above to explain why a long variable is less likely to experience the same ‘over the memory limit’ problems experiences in #3 above.

Explain why Long variable are less likely to go ‘over the memory limit’ here

## Perform subtraction on the microcontroller

Modify the [Example Sketch 1](#30j0zll) so that it becomes an example of [subtraction on the Arduino](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.883ooh5nq6o0). Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot as evidence of your success.

Screenshot goes here

How did you need to modify the entire program to make it subtract? How did you know to do this?

Provide complete responses to these questions here

## Modified subtraction on the microcontroller

Type the program from task #5 above so that the Arduino does subtraction but change it so that you end up with a negative number. Take a screenshot of this working, and make sure that the program and serial monitor results are visible to show your success.

Screenshot goes here

1) Explain what role the variable ‘c’ plays in the [example sketch 1](#30j0zll) above, and 2) if the variable ‘c’ is absolutely necessary to get these same results from the mathematical operations you’re using here.

Provide complete responses to these prompts here

## Perform multiplication on the microcontroller

Type the program above so that the [Arduino does multiplication](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.883ooh5nq6o0). Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot to show your success. *\*\*Be careful of overflowing the variable type when you multiply -- products get big much faster....*

Screenshot goes here

How do you know that the screenshot you included above is an example of this calculation going ‘over the memory limit’ or not going ‘over the memory limit’ of the Arduino? What would you look for?

Provide complete responses to these questions here

## Perform modulo on the microcontroller

Type the program above so that [the Arduino does finds the modulo of the inputs](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.883ooh5nq6o0). Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot.

Screenshot goes here

Explain what modulo is, in plain english. You may research an answer, so long as you understand it well.

Provide complete responses to these questions here

## Perform division on the microcontroller

Type the program above so that [the Arduino does division](https://docs.google.com/document/d/1kGjuOrPaoF2TtsOMb4omhPFz-2F4GSLLcFJxVTIabuY/edit#bookmark=id.883ooh5nq6o0). Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot. *\*\*Be careful that the division gives you the correct answer >>* ***you may need to adjust the input values.***

Screenshot goes here

How do you know that the screenshot you included above is an example of this calculation working as it should in math class? What would you look for to be certain that it worked? How could you fix it if it does not work equally as well as in a pure math class?

Provide complete responses to these questions here

## Perform ‘floating’ division on the microcontroller

Type the program above so that the Arduino does division. Take a screenshot of this working, and make sure that the program and serial monitor results are visible in the screenshot. *\*\*Be careful that the division gives you the correct answer >>* ***this time make the variables all ‘float’s.***

Screenshot goes here

How do you know that the screenshot you included above is an example of this calculation working as it should in math class? What would you look for to be certain that it worked? How could you fix it if it does not work equally as well as in a pure math class?

Provide complete responses to these questions here

# Reflection

***Reflect on what just happened. What did you see that you expected? Didn’t expect? What else did you learn by doing?***