

# Lab Assignment 0: Environment Setup

## Objective

**This lab is worth 5% of the course mark.**

The lab, despite its simplicity with regard to coding, it requires you to get your environment for the course, work with git revision control, and perform a unit testing.

Follow the lab steps to make sure you are able to get your environment working and that you are familiar with it.

## Helpful References

Read the provided documents under Lab0 in A2L. They walk you through the git revision control basics as well as the unit testing fundamentals.

In 2SH4 labs, we will use *Eclipse*. Using Eclipse is not mandatory, it just helps you organize your code and run it using a graphical user interface. Otherwise, you can easily run things from the terminal. We also provide a *Makefile* that does the compilation process and running code (Example on how to use this is in Section 4).

Please note also that you can use your own setup/IDE if you wish. However, given the too many possible combinations of tools and OSes, we highly recommend that you use the environment discussed in this manual based on Eclipse. TAs will only provide support for issues you face related to this environment.

We now discuss how to setup your environment in a step-by-step guide.

## Create Github Account

Go to [github.com](https://github.com) and signup using your McMaster email address. Please note that you MUST use your mcmaster email address account throughout these labs.

## Environment Setup:

### 1) Obtaining Eclipse

- 1) Download Eclipse Installer from here: <https://www.eclipse.org/downloads/packages/installer>

- 2) Launch it and follow the steps. For the “Select the package to install” choose “Eclipse IDE for C/C++ Developers” and continue the installation process.

## 2) Obtaining Java VM

Eclipse IDE is based on Java VM, even for C/C++. So you need to make sure your system has a Java VM. Most probably, you already have it installed as it is required for many software. However, assuming you do not have it installed follow the following steps.

- 1) Go to the Java JDK download page and choose the one suitable for your OS:  
<https://www.oracle.com/java/technologies/javase-jdk14-downloads.html>
- 2) After downloading the binary, run it to complete the installation process. No special instructions are needed here.

## 3) Obtaining a C toolchain/Compiler

You might reasonably expect that once you installed something named “Eclipse for C/C++ Development” you would be able to develop C (or C++) programs with it. Unfortunately, that isn’t the case. In fact, it isn’t the case for Java development either. For Java development, you need to install Java. For C development you need to install a C toolchain. Here are the steps to do so:

### 3.1) Installing Developer Tools on a Mac

Download the XCode IDE from the Apple App Store. This is a huge package that supports development for macOS and iOS, but it also includes all the programs that you need for C development using Eclipse.

### 3.2) Installing MinGW on Windows

Follow the install guide named MINGW\_C\_Install provided in the lab folder in A2L.

- Ensure on the left that Basic Setup is highlighted. Click the three boxes indicated below: mingw32-base, mingw32-gcc=g++, msys-base.
- After clicking each, select Mark for selection.

### 4) Obtaining Git Support

EGit is already included in most recent Eclipse versions. We will install some other tools to get git terminal support (we will be using git in terminal).

Follow these steps to do so,

1. Install git <https://git-scm.com/download>
2. Install TM terminal from Eclipse marketplace: <https://marketplace.eclipse.org/content/tm-terminal>
3. After that, in Eclipse: go to Window -> Preferences -> Terminal -> Local
4. add a entry with full path where Git Bash executable was installed. For example, mine is at:  
C:\Program Files\Git\bin\sh.exe
5. Close and re-open eclipse
6. To test this, open a terminal as follows: Windows-> Show View -> Terminal
7. Write "git status" and see if you get something meaningful like "not a git repository" that means git is correctly installed.

## Importing the Starter Code

- 1) go to and sign in using your previously generated account
- 2) click the following link and accept the invitation: <https://classroom.github.com/a/h8FMWB9M>  
this will create your lab repository. Once done, you should have a repo named something like: <https://github.com/Fanus-Courses/lab0-username> where username is your github's user name (most likely it is also your macid since you used your McMaster email address to sign up).
- 3) Now, you need to import this repository to your local machine to start developing the lab using Eclipse.
  - a. Open the Eclipse Import wizard (e.g. File => Import),
  - b. select Git => Projects from Git and click *Next*. Note: new Eclipse versions have another option " Projects from Git (with smart import) , make sure you do NOT pick this option.
  - c. Select "Clone URI" and click next.
  - d. Entering the URI (this is the repository generated link for you (e.g.: <https://github.com/Fanus-Courses/lab0-username>) and this will automatically fill some fields.
  - e. Complete any other required fields and hit *Next*.
  - f. For the Local Destination screen, select your destination directory. *As a good practice for this course, keep a folder named "2SH4-Labs", where you will checkout all the repositories for all labs and do all the developments there. Also, if possible, create this folder in a drive other than your system drive (for instance, use any drive other than C:\).*
  - g. Click *Next*, and this will checkout the repo to "2SH4-Labs\lab0-username" folder.

- h. For the window “Select a wizard to use for importing projects”, select “import using the New Project wizard” and hit finish.
- i. In the next screen, select “C Project” and Next.
- j. Then enter a Project name, untick the use default location option and browse to the folder you created for the project from previous steps (e.g. 2SH4-Labs\lab0-username). Then, select “Empty Project” from “Executable” and select your toolchain (it is the one we installed previously, e.g. for Windows it is “MinGW GCC”. Then click “Next”.
- k. For select configurations, leave as default, and click “finish”.
- l. You are all set and the project should load to your Eclipse IDE in the project explorer.

## Developing the Lab

- This lab main purpose is to introduce you to the environment including Eclipse, git, and unit testing.
- The folder structure is as follows:
  - The test library files: *CuTest.c*, *CuTest.h*
  - The test suite files (those are the ones we edit to create tests): *AllTests.c*, *testCases.c*
  - The lab header file: *Questions.h*
  - The lab source files: for lab0 it is only one file *Question1.c*
  - A *Makefile* used for test purposes.
- **You must NOT touch the *Makefile* nor the *AllTests.c* files. For the *testCases.c*, you are only allowed to add more test cases, but you must NOT touch the already existing tests. When grading the lab, the environment checks for any changes in these files. Upon violating these rules, you will get zero for violating academic integrity.**


### 1. Lab Question

- 1) First, you are asked to go through the *Questions.h*, *Question1.c*, and *testCases.c* and make sure you understand them.
- 2) Locate the main function.
- 3) Complete the implementation of the function
  - `int addFunction(int input1, int input2`
 in the *Question1.c* file, which takes two input integer parameters, adds them, and returns the sum.
- 4) Add another function `void TestQ1_3(CuTest *tc)` to the *testCases.c* file to test your implementation. Think in some useful tests (such as adding negative integers?)
- 5) Once done, you need to compile your code. This is done by selecting the project from the project explorer, and then click Project -> Build Project.

- 6) Check if the project is built successfully. You can see the building steps and result in the Console window, if Console Window is not shown, you can show it using Window -> Show View -> Console)
- 7) Now, you can run your project: right click the project in the project explorer and select Run As->Local C/C++ Application.
- 8) Observe the output in the Console Window. The unit testing framework will detail to you whether your program passed all tests, and for failed ones, it displays the expected vs your actual output. Make sure you pass all the provided tests. In our end, we have some additional testing that are not provided to you to further stress the correctness of your program. So, make sure you think carefully about covering all possible scenarios.

## Pushing Changes to the Github Repo

Once you are done developing your lab and ready to push changes to the repository, follow these steps:

- 1) Open a terminal window. If it is not already shown, show it using Window -> Show View -> Terminal). In the terminal window, click the small blue computer icon to open a new terminal: 
- 2) Navigate to the project folder using `cd` command.
- 3) Type `git status`, this should give you the files you changed.
- 4) Type `git add *`, this will add files to the commit list
- 5) Type `git commit -m "message"`, where message is a meaningful message of your choice.
- 6) Finally, `git push`, will push these changes to the repository.
- 7) Note that in your github repository page you can see all your commits and their status, i.e. whether they passed testing or not.

**Note that: only the last commit is used in our environment for grading.**

## Good Version Control Practice

This lab might not be very involved and you can finish it in one setting. However, in remaining labs (and surely, in big real projects), developing is done over a period of time. It is very important

that you keep your code consistent(think of the case that you work in a project with other people and you all edit same code base), to do so, follow these easy steps:

- 1) Each time you start working in the project/lab, do a “git pull”, this will update your local repository with any changes conducted on the remote “shared” repository.
- 2) Each time before closing your session (e.g. closing Eclipse), make sure you push your updated following the steps in the previous section. In fact, even within the same session, if you make some big changes (e.g. finishing one function/question), make sure you push that.

## Lab Deadline

The deadline for lab0 is Sept 18. Please note that this is a programmed deadline in the environment, so make sure you submit in time since you will not be able to submit after that dictated deadline.