SYSC3310 Lab 1 – Running a program on the Lab equipment using Keil

Fall 2018

By the end of this lab, you will be able to write, build and download a program onto the MSP432P401R Launchpad within the Keil Integrated Development Environment (IDE).

- It is a long lab so you will be permitted to demonstrate the final task at the start of Lab 2. This is an exception – **All other labs must be demonstrated by the end of your lab period.**

Objectives:

- Deadline: Finding and registering your Lab Partner
- [Optional] Preparing the development environment on one's own laptop.
- First experience within the development environment on building and downloading a program (on the lab machine or on your own machine)
- Learning to use TI's Resource Explorer
- Execution of standard test codes that can be used repeatedly to verify that hardware is working.
- Writing your first C program from scratch.

Equipment:

- [Suggestion] Bring your laptop to the lab to prepare it for working at home during the term.
- **MSP432 P401R LaunchPad Development Kit**
  - **Your kit contains 3 boards, but you only need the Launchpad for this lab**
  - **You do not need the hardware at all until Part E**
- KEIL uVision, MDK-Lite (32KB) Edition
- [Alternative-not described] Texas Instrument (TI) Code Composer Studio (CCS)
- [Alternative-not described] Texas Instrument (TI) Code Composer Studio – Cloud Version

References and Reading Material

- http://users.ece.utexas.edu/~valvano/Volume1/uvision/ - Valvano's page for installing Keil uVision
- http://users.ece.utexas.edu/~valvano/arm/ - Valvano's Book Companion Site
- CMSIS-DAP Debugger Manual: http://www.keil.com/support/man/docs/dapdebug/dapdebug_ctx_trace.htm
- Logic Analyzer Setup: http://www.stmcu.org/module/forum/forum.php?mod=viewthread&tid=602588&fromuid=394920

From Valvano's Email
His Suggestion: http://edx-org-utaustinx.s3.amazonaws.com/UT601x/RTOSdownload.html
Other Reading: http://users.ece.utexas.edu/~valvano/arm/downloadmsp432.html

Part A: Lab Partners

Labs will be done in pairs, the same pair throughout the term. This lab is your last chance for finding – and registering – your lab partner. If you do not have a lab partner, contact the TA for help in finding another such student.

Please read the Course Outline as well as posts on the course's CULearn webpage for full details on lab partners, including marking and implications of absences by one partner.

Part B: Lab Equipment

Lab equipment will be your (pair's) responsibility for the term. Equipment will be given to you (and your partner) – for a deposit. You will be responsible for keeping the equipment safe and in working order, and for bringing it to each lab period.

Please read the Course Outline as well as posts on the course's CULearn webpage for full details on lab equipment, including procedures for picking up and dropping off the equipment.

If your lab period is during normal office hours, you are welcome to use this opportunity to get your equipment from the Systems office ME4456. If the office is closed, you will have to do it on your own time, before the next lab.

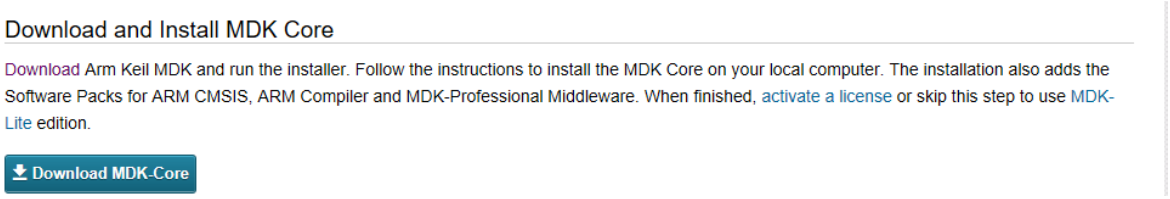
Most of this first lab can be completed without the lab equipment. Please proceed, even if you do not have the equipment. **In Part E onwards, you will use the Launchpad board only.**

Part C: Preparing the Development Environment

This part of the lab may be done at home, on your own time, so that you can work on future labs at home on your laptop. You may skip this step if you are using the lab machines.

The objective is to download and install the **Keil** development environment and all supporting libraries and sample programs for both the MSP432P401R **Launchpad** and Educational Boosterpack **MKII**.

1. Download KEIL uVision MDK-Arm – Development Environment for Cortex and Arm devices (IDE)
   URL: http://www2.keil.com/mdk5/install/



   - The file is MDK525.exe and is 837 MB. Depending on your internet connection, the download takes 20-30 minutes
   - You will need to register with ARM, and use the **LITE version** (Limited to programs of 32 KB)
     - When registering, there is no need to enter the type of board being used

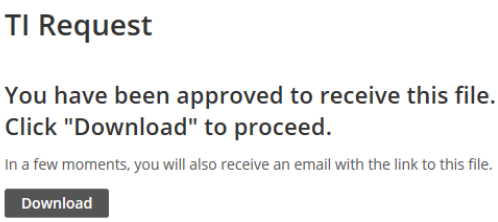
    - If you don't want to use your personal information, please use the departmental information (address, phone number); available by searching on the web.
- After downloading, run the executable file to install the program.
- Use the default location when installing. Pay attention in which folder Keil is installed.
    - Default Location: c:\Keil_v5

2. Download and install TExaS - TExaS is a suite of drivers and sample programs written by Jonathon Valvano that we will use to test our hardware. The suite contains many projects for both Keil as well as Code Composer Suite (CCS – which we are not using) as well as projects using a Real-Time Operating System (RTOS); we will use only those for Keil and those without an operating system.

   URL : http://edx-org-utaustinx.s3.amazonaws.com/UT601x/RTOSdownload.html

   - Scroll down to subtitle Step 2) TExaS Lab Graders for Labs 1 to 5
   - Click on the link: Download Texas Labs 1,2,3,4,5
   - The file is RTBN_Install.exe.
   - After downloading, run the executable file to install the suite "under" the Keil program
       - Using Windows explorer, ensure that a folder was put in C:\Keil_v5\MSP432ValvanoWare

3. Download and install Windows drivers for the MSP432 LaunchPad board
   - It is recommended that to always to download the latest drivers.
   - You will need to create a TI account, but it is free; however, as part of the procedure, you will have to sign an U.S. Government export agreement.
       - Use your Carleton University credentials
       - If the Postal Code is an issue, please enter: 32259. If that does not work, try Carleton's postal code (K1S 5B6) – with space and with capital letters.
   - To begin the download procedure: Click on this URL.
   - The file is ti_emupack_setup6.0.4073.win_32.exe.



   - After downloading, run the executable to install the drivers.
       - As before, when given the option, install the drivers "under" the folder in which you installed Keil (Default: c:\Keil_v5) (TBD: When re-tried, default was c:\ti – It worked too)

Part D: **First-Time** Use of the Development Environment

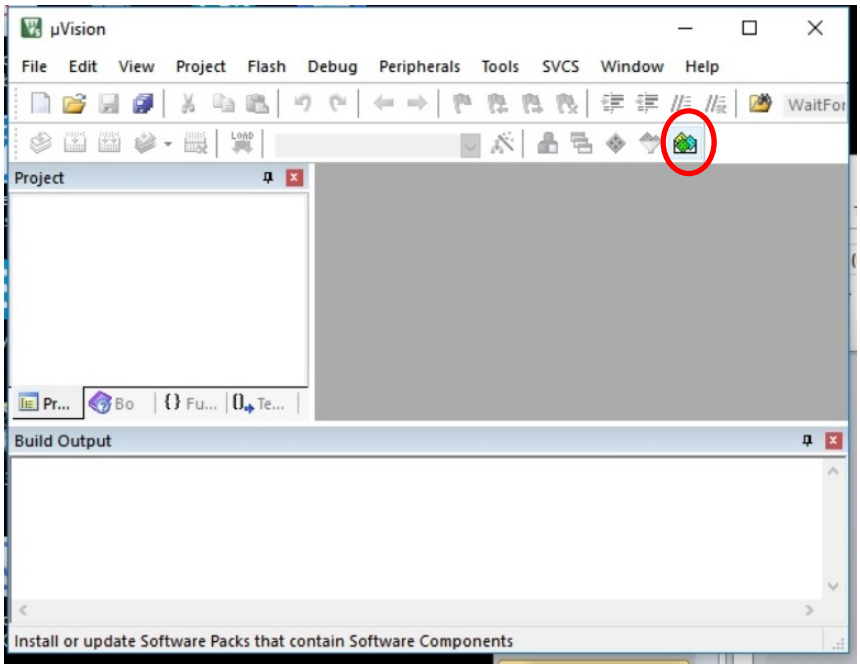
This part – and all remaining parts –of the lab may be completed on the lab machine and/or on your own machine. Even if you are using the lab machine with the software installed, this final "configuration" stage may need to be followed, if it is the first time that someone has used the software on that machine.

The objective is to "configure" the Keil development environment to use the MSP432P401R Launchpad. Configuration is needed because Keil (and most other embedded IDEs) work not just for one version of a computer but for many, many versions of the computer. Configuration will identify for which computer that you want to develop programs – the MSP432P401R (Red).

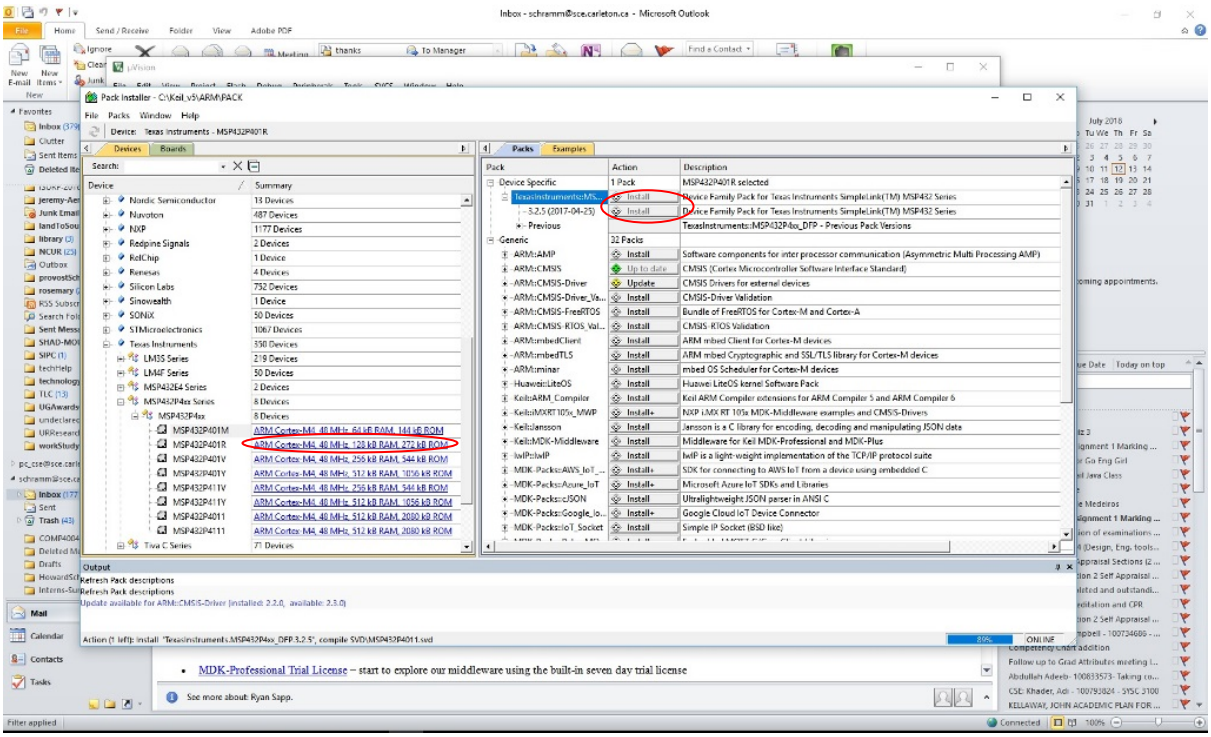
During this stage, you will also learn to navigate both the installed sample programs and the Texas Instruments **TI Resource Explorer**. In doing so, you will run your first program.

1. Read this Getting Started reference, scrolling down to Install Software Packs.
    - Make sure that you know how to find the following: **Devices** tab, **Packs** tab, **Examples** tab.

2. Open up KEIL. If it is the first time that the program has been run, a special dialog window will be displayed. If not, please click the **Pack Installer** icon (shown in the red circle below)



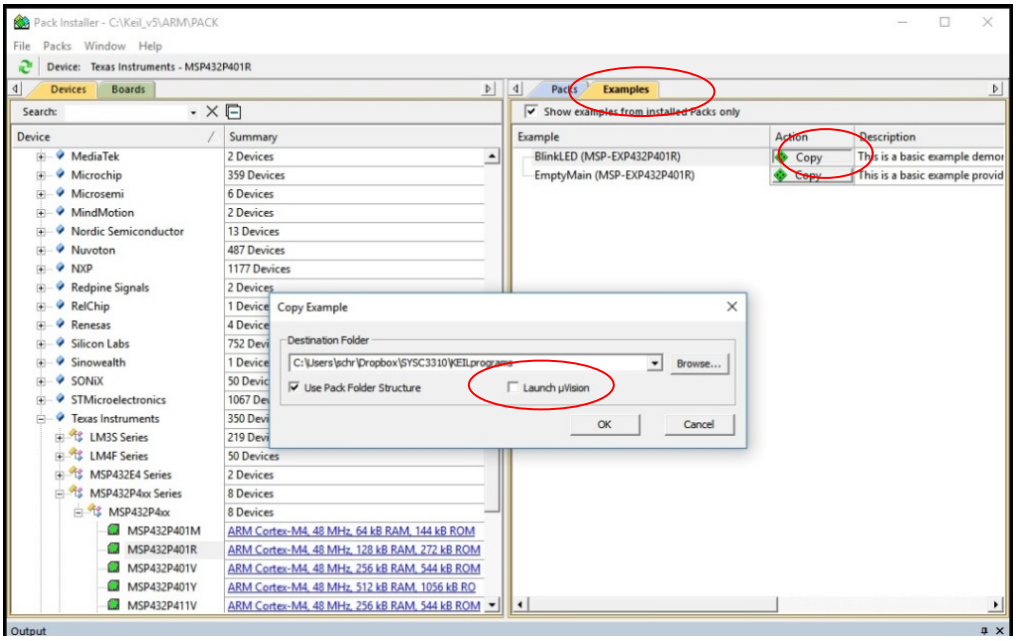
3. Using both the **Getting Started** reference in Step 1, and the image below, install the packages using:
    - On the **Devices** tab: ARM Cortex-M4 40 MHz, 64kB Ram, 144 kB ROM
    - Click on Texas Instruments -> MSP432P4xxSeries->MSP432P4xx->MSP432P401R
    - On the **Packs** tab: Device Specific: Texas Instruments
        - Click on **Install** for Device Family Pack for TI SimpleLink™ MSP432 Series
        - Click on **Install** for 3.25 (2017-04-25) for Device Family Pack for TI SimpleLink™ MSP432 Series
        - For now, don't install others packages, but if you are interested, you can come back here and explore more later in/after this course; for example, to use an RT-OS.
        - Click **OK** to apply, but do not close this **Install-Packages** window. We still need it.

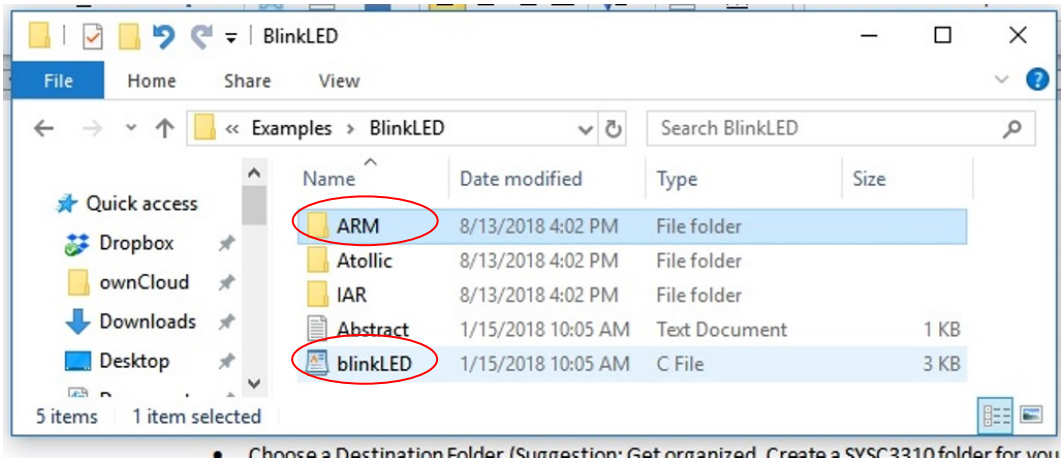


Tip: Ultimately, we are going to be writing C programs, stored in a .c file; but when working within an IDE such as Keil or Code Composer, you also work within projects. A project is a set of folders and files – including the C program file (.c) – that work together to build an executable for a particular computer.

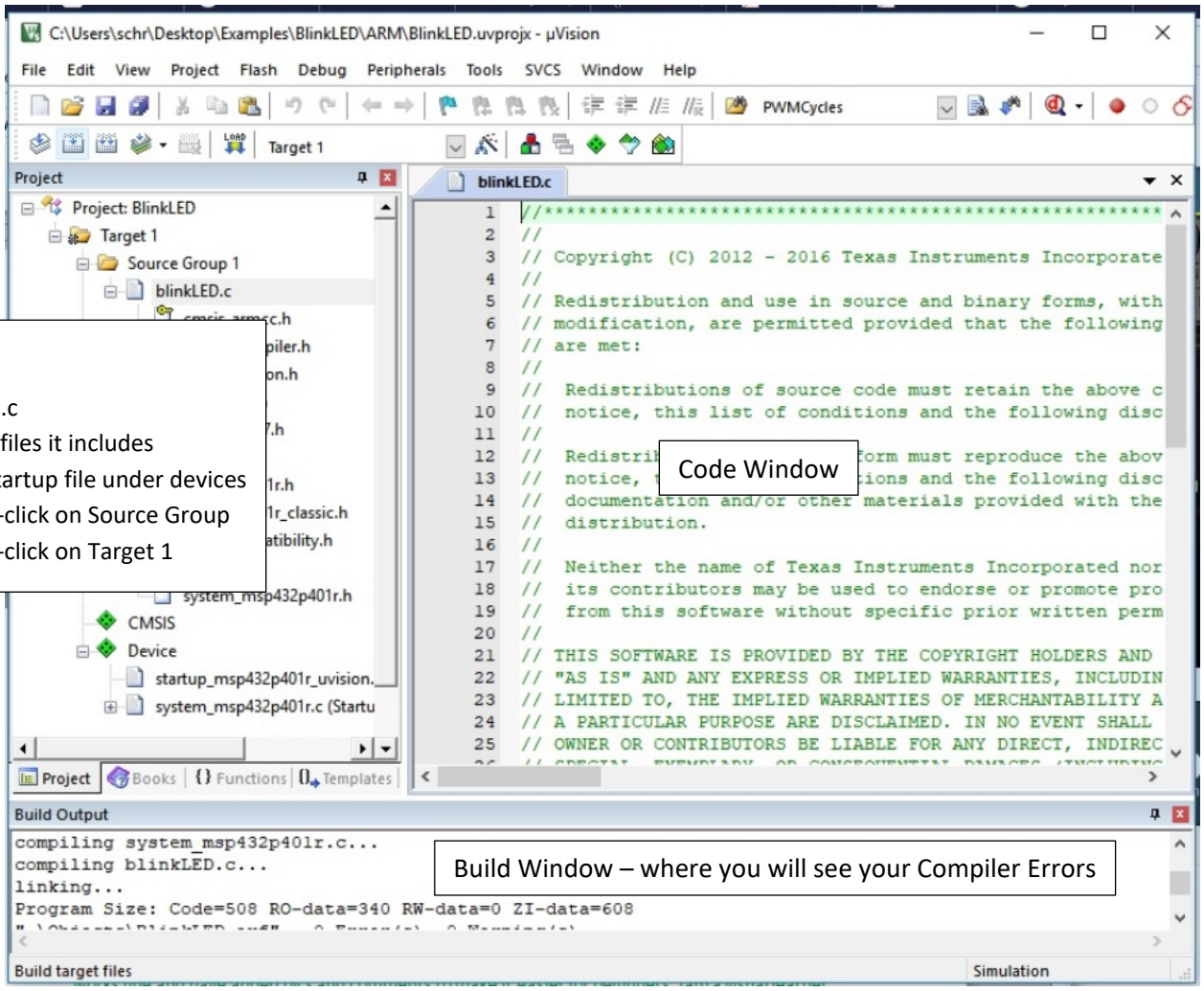
As we progress through the labs, pay attention to which word is used: Project or Program.

4. We want to verify the installation by running an Example project. Get a copy of the **BlinkLED** example program (See Figure below)
    - Still within the **Pack Installer**, navigate to the **Examples** tab, instead of the **Packs** tab.
        - Alternative: Learn where the files are actually stored and access them directly. For example, the BlinkLED project is stored in C:\Keil_v5\ARM\PACK\TexasInstruments\MSP432P4xx_DFP\3.2.5\Examples
    - Select **Copy** for the BlinkLED program.
    - Choose a **Destination Folder**
        - Suggestion: Get organized now! Create a SYSC3310 folder for all your work this term.
    - You can either **click** or **un-click** the Checkbox for "Launch uVision". It will work either way.
        - If you don't click it, you can learn what-and-where files are stored in a Keil project.
            - Using Windows File Explorer, navigate to the **Destination Folde**r you chose in the previous step. (See Figure below, on next page)
            - Continue navigating into the subfolders down to : Examples\BlinkLED
                a. **Examples\BlinkLED** is your project folder
                b. Here you will see a file called **blinkLED.c**. This is your program
                c. You will also see multiple subfolders, for different machine architectures. All three architectures reference the same folder. They just build it differently, depending on the memory maps and instruction formats of the particular machine.
                d. We are using an ARM so navigate once deeper into this folder where you will find a file called **BlinkLED** (uVision5 Project file) (with a green-and-white icon). This is your project file.
                e. Double-click on the project file to open Keil.





5. Take a moment to look at the Keil IDE. It opens in the Code Perspective, shown below.

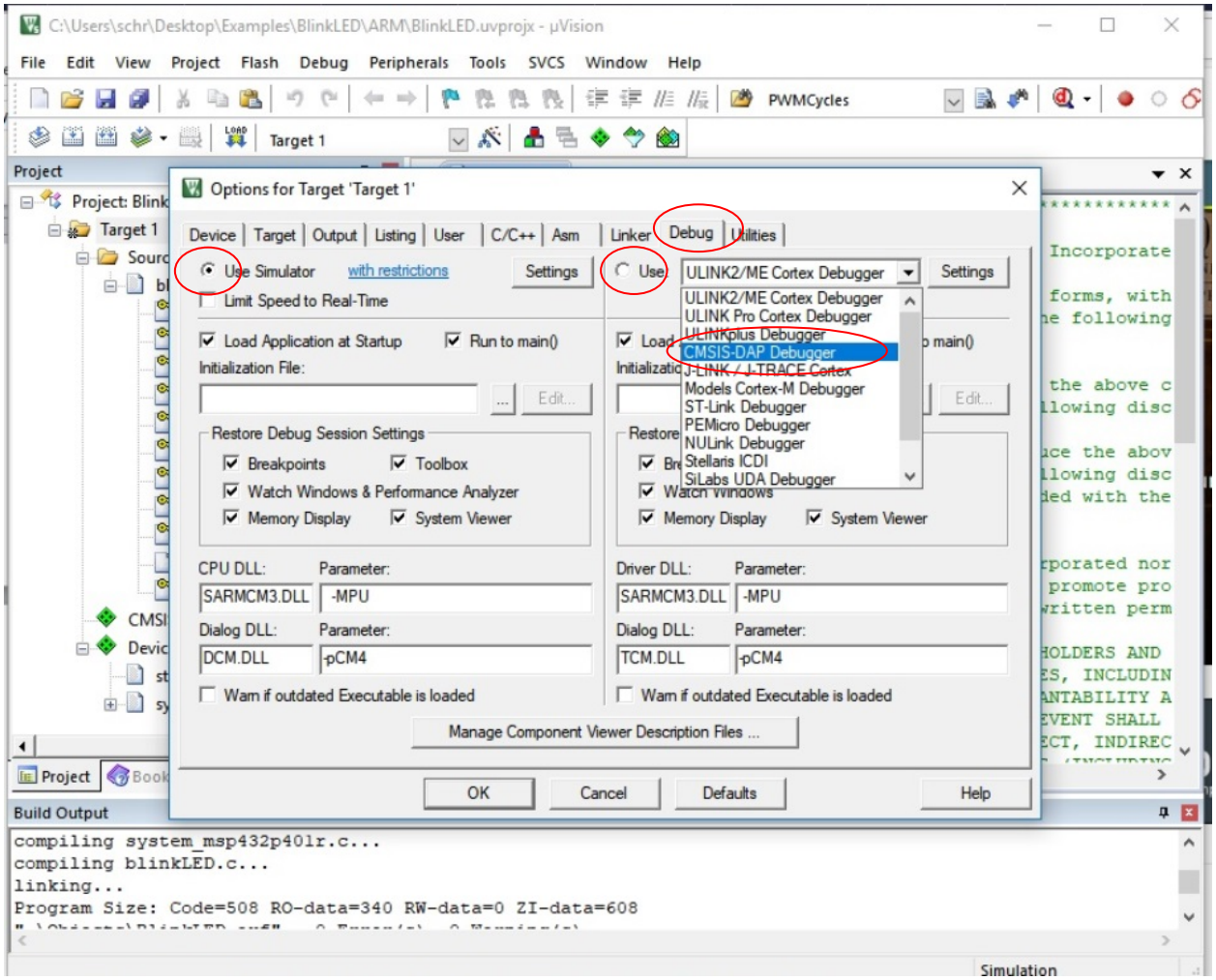


Project Window

- Find blinkLED.c
- See all the .h files it includes
- Look at the startup file under devices
- For fun, right-click on Source Group
- For fun, right-click on Target 1

6. Build the project. Select Project->Build Target

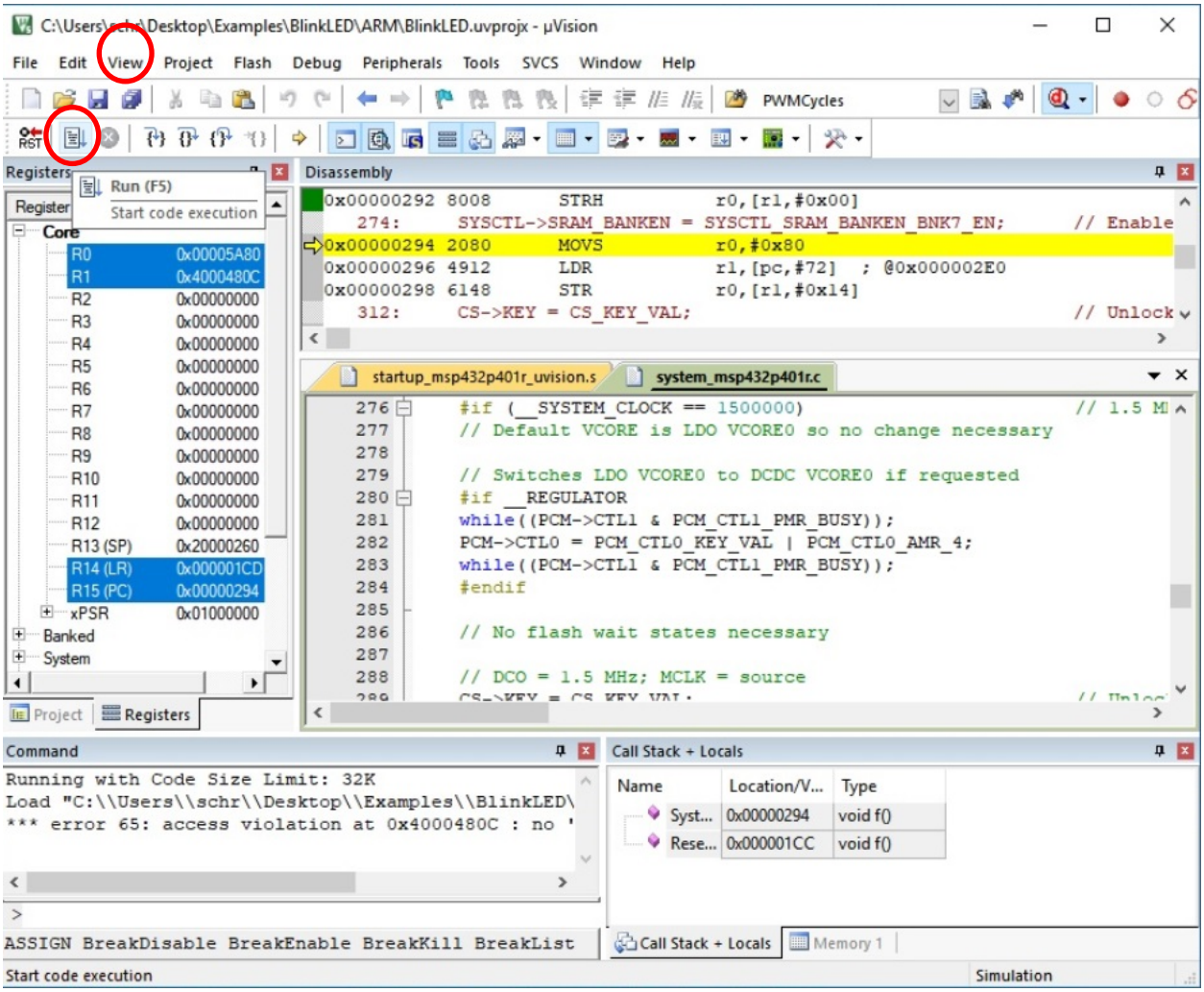
7. Configure the Debugger for the project. Select **Project-> Options for Target 'Target 1'**
    - **LEARN THIS STEP. You will have to do it every time you create a new project!**
    - See Figure below
    - Select the **Debug Tab**
    - The Debug Tab has two columns where you have to choose one or the other.
        - LEFT column: **Use Simulator**
            - A simulator is a software program that runs your program on a software-implementation of our computer. We will generally NOT be using the simulator but in this first lab, if you don't have your hardware yet, it is a great replacement! Check this box.
        - RIGHT column: **Use Debugger** (i.e. with real hardware)
            - This is the option that you will generally pick for the remainder of the term, every time your create a new project.
            - Select the **CMSIS-DAP Debugger**
    - Click **OK**



8. It is finally time to connect up your Hardware: Just the Launchpad and the USB cable that will connect it to the host machine (either the lab machine or your laptop).
    - A green LED on the left side should turn on once there is power.

9. Run your program
    - Select **Debug-> Start/Stop Debug Session**
        - If you watch the Build Window closely, you will see messages and a progress bar that shows the downloading of your program to the hardware.
    - A new perspective is loaded, called the Debug Perspective (See Figure below)
        - Some default VIEWs are shown: Registers for low-level debugging, Assembly translation of your C code, your C code itself, the Call Stack showing local variables (if any)
        - As time goes by, explore other VIEWs by selecting **View** on the top menu.

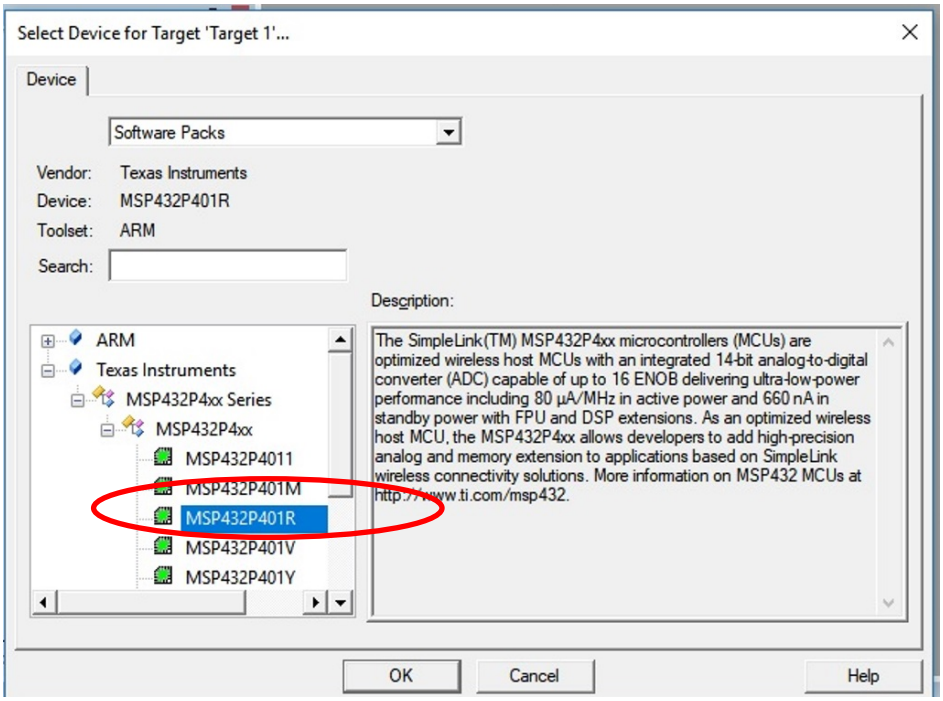
- At this point, the program is downloaded (into the hardware or the simulator). It is NOT running! You actually have to launch the program by clicking on **Project->Debug** or the **Run** icon (See Red Circle in the Figure below)
    - If you are running on real hardware, you should see a red LED blink.
    - If you are running on the simulator, sorry but you'll see nothing, but you will have learned all the steps in running an existing project.



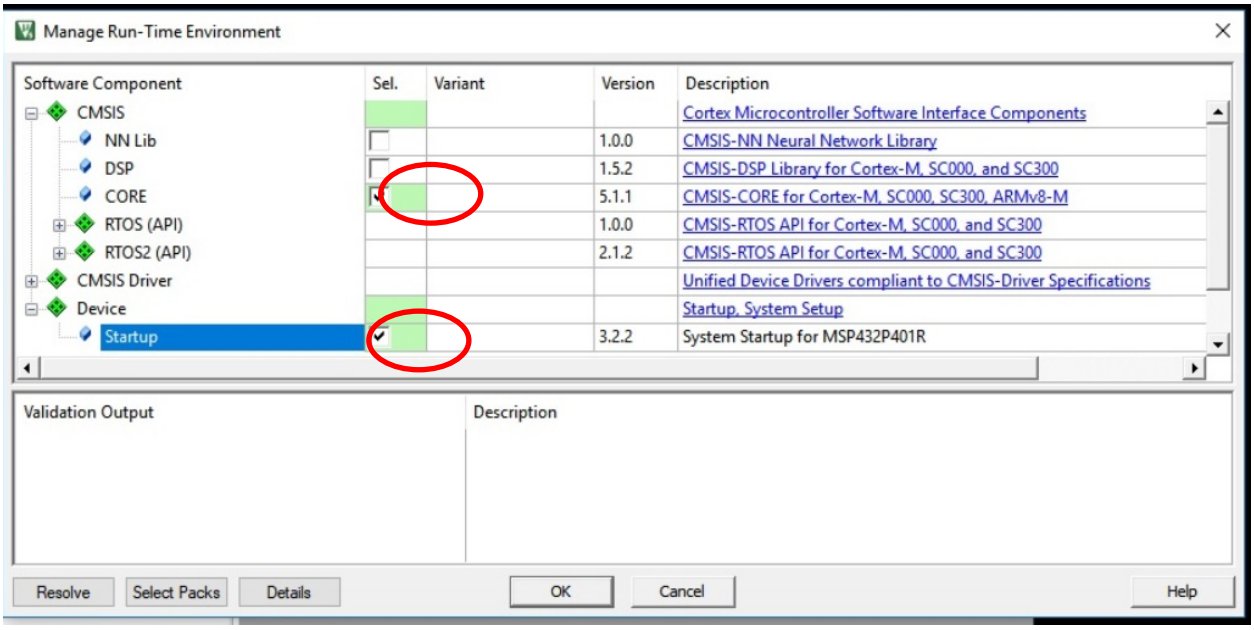
Did you know? You can edit the code in the debug perspective, but be mindful that this new code is not running on your machine. To do so, you must actually stop the debug session – going back to the Code Perspective – re-build your target and re-start the Debug session (to re-download the new code).

Part F: Creating your First Project

1. You've run an existing project. It is time to create your own project from scratch. Keil has an oddity in project creation that makes it vital that you first watch this fabulous video. In fact, watch it, and then rewind it and play it while proceed with the next step.
    - The video shows the creation of a project. There are some odd-yet-vital steps about creating a folder, before creating the project.
    - The video works with an assembly program. We will be programming in C. It is still very instructive.
    - URL: https://www.youtube.com/watch?v=i9zwf_W2e58
2. Open a new instance of Keil, or close the previous project.
    - Project->Close Project.
    - Do NOT create a new project, yet.
3. Using the video as a guide, create an new project within Keil
    - As shown in the video, **before** creating a project inside Keil, you must first create a folder outside Keil (using Windows File Explorer)
        - You can choose where to store your project's workspace. Again, I encourage you to get organized and create a SYSC3310 folder (and subfolders for each lab).
        - Call the folder **Lab1**
    - Within Keil, create a new project
        - Select **Project->New uVision Project**
        - Browse to the folder that you just created
        - Enter in the name of your project: **Lab1**
        - Click **OK**
        - A new dialog should pop open called the **Device** tab (See Figure below, on next page)
            - Navigate through the **Texas Instruments** devices until you find **MSP432P401R**.
            - Click **OK**



- Another new dialog should pop up called **Manage Run-Time Environment** (See Figure below)
    - Select **CMSIS-CORE for Cortex-M, SC000, SC300, ARMv8-M**
    - Select **System Startup for MSP432P401R**
    - Select **OK**



You have created your first project. You will repeat this procedure for every lab.

4. It is time to write your first program within your first project.
    - Again, you can refer to the video, although simply replace C files whenever assembly files are used.
    - Right-click on **Source Group 1-> Add New Item …**
    - Select **C File (.c)**
    - Enter in the name of the file: **main.c**
    - Enter in the minimal code below

```
#include <stdint.h>

int main(void) {

      return 0;
}
```

5. Build and run your tiny program. It won't do anything yet, but you re-inforce the steps that you need to take to build a project. In particular, I suspect that you will have a problem downloading your code to your hardware! Why? Because we have not configured our Debugger! Refer back to Part E, Step 7.

Part G: Writing your First Program

It's time to take the training wheels off and let you do some programming by yourself. We haven't covered anything in our lectures yet, so instead of using the hardware, we will simply write an ordinary data-processing C program. Credit goes to Jonathan Valvano, Introduction to the MSP432 Microcontroller, Volume I, page 126.

Problem: Write a random number generator. The random number generator creates a pseudo random sequence of numbers. This means the generator will return a unique 32-but number for the first 2^{32} calls. After that it will loops through that sequence over and over. Let *N* be any constant less than 32 bits, and let *M* be $2^{32}/N$ (rounded up). If we calculate *n* = (random/M) then *n* will be a random number from 0 to *N*-1. Then random number generator is fair: for every calculation of n, the chances it is 0 to *N*-1 are equally likely. Then random number generator is random: if I know prior determinations of *n*, the chance of the next generator of *n* is still equally likely to be 0 to *N*-1.

Solution The solution uses a linear congruential generator (LCG).

$M_{n+1} = (a * M_n + c) \bmod m$

M_0 is the seed which is the first number in the sequence, set to 1 (but can be reset)
a is the multiplier, set to 1664525
c is the increment, set to 1013904223
And $m = 2^{32}$.

The program will have two functions (other than main): (1) An initialiser called **randomInit**() that sets the seed to the value of its 32-bit number argument, and (2) **random**() which returns a 32-bit random number whenever called.

Furthermore, the main() program shall make use of the random number generator.

> *Because we cannot print out the results(Did you notice? There is no screen!) we are going to have to simply store our results in memory and then use the debugger to inspect the memory to verify the results.*

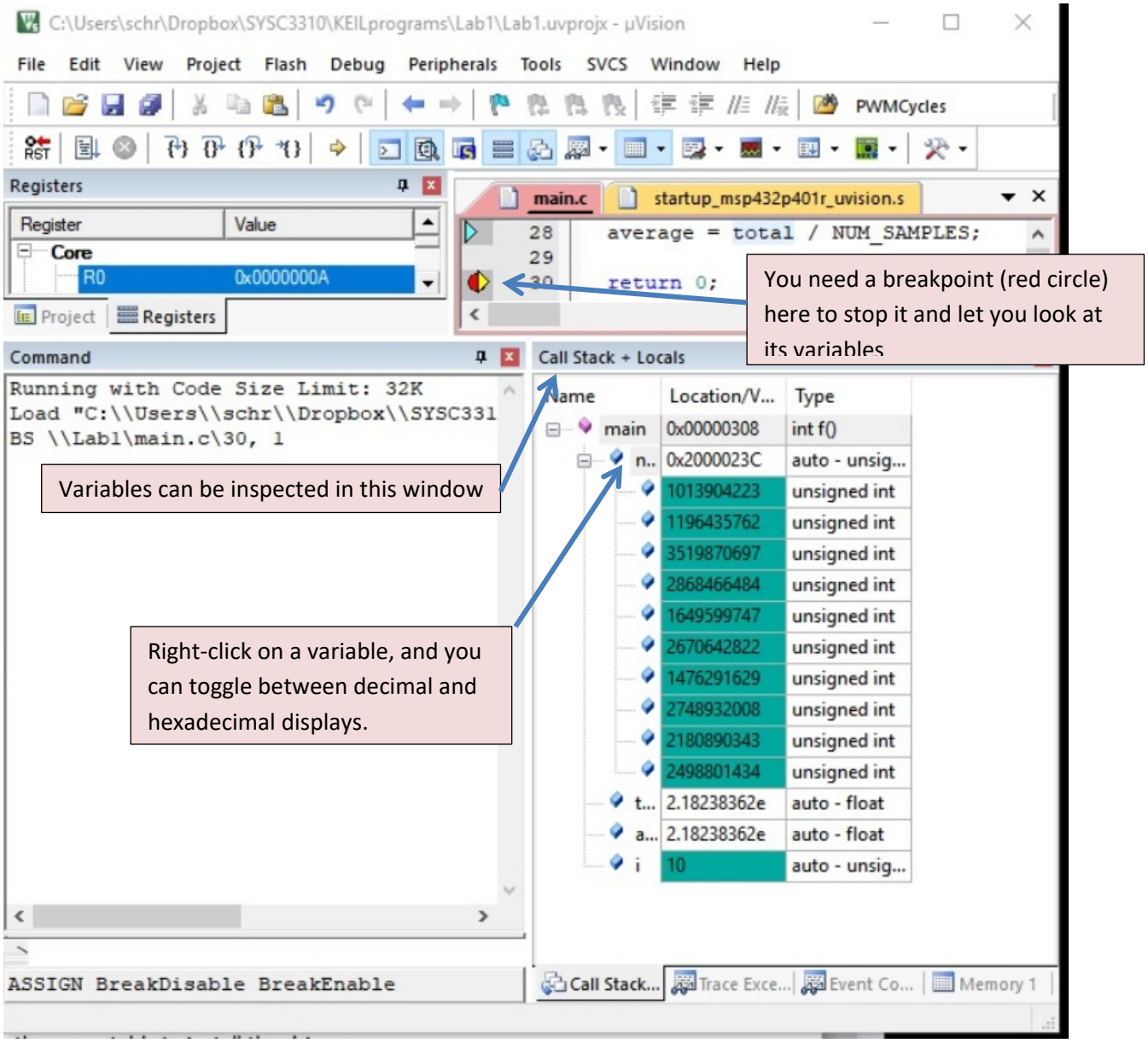
The main program shall:

1. Call **randomInit**() to initialize the seed to a value equal to the current time, expressed in seconds from the beginning of the this year.
2. Call **random**() 1000 times to generate a statistical sampling of random numbers. Each random number must be stored in a local array, and the average of all random numbers generated must be calculated and stored in a separate local variable.

You must write the program and demonstrate its results to the TA (See Figure below, on the next page)

Tips – The embedded version of C may be different from the versions you used in other courses.

- Global variables are defined outside the scope of any function (outside of {})
- Variables and functions must be declared before they are referenced.
- For-loop variables cannot be declared within the floor loop; they must be declared at the top of the function with the rest of the variables.
- You must use the types uint32_t, uint16_t and float instead of just ints.



Marking Scheme:

Demonstration: Separate marks for each student

- Demonstration of Part G
- Both partners must then demonstrate the creation of a new project (Repeating Part F without looking at the instructions)

Inspection: 1 mark each

1. Functions declared as described
2. Random() must implement the math as described
3. Main() must have an array filled by multiple calls to random()
4. Overall Style

Overall marks for style (To be used for all labs in this course):

- Comments, indentation, and well-named functions and variables
- Removal of all extra code (no commented out sections, no unused code leftover from some other example)