# ICT283 Lab 11 Assessed exercise 3 (last one)

# 

Like the two assessed exercises that had to be demonstrated in person, this exercise is part of the personalised assessment component and needs to be demonstrated in person.

**Objectives:**

* The purpose of this demonstration/submission is to ensure that you are working on the assignment yourself and that you are adhering to the following good practices:
  1. Menu options have been added/updated to cater for assignment 2. The work does not have to be complete, but it must build (compile and link) and run, and you can explain/defend what you did in-person to your tutor. Menu options should be there but non-working parts ***stubbed***. The stubbed portion can just print to screen that it has been called.
  2. How you have been keeping backups as these are necessary so that you do not lose all your work if something goes wrong.
  3. Be aware of the minimum requirements of the assignment, as not meeting the minimum requirements would result in failing the assignment.

**All students** demonstrate/defend the work to their tutor during the next lab session. Typically, this will be during the last teaching week. You can demonstrate earlier if you are ready. For this demonstration, it is not expected that you have completed the entire assignment. You would be showing your progress on it and explaining your thinking about the approaches you have taken. Explanation of “**why**” you have done something is important.

Can any approach from Topic 11 PowerPoint notes (*Lec-31.ppt*) be used to better balance the tree without actually changing the tree from lab 10.

You may be asked to write out design and code from scratch in notepad (not an IDE) and/or on a whiteboard. Then you will need to explain how your code works. Like Assignment 1, there are no marks allocated to assignment 2 without the personal demonstration. This practice of doing from scratch without tools like an IDE is also preparation for the final exam.

It is not possible to demonstrate after the last teaching week.

## Exercise (assessed)

Please ensure that the data files for assignment 2 are read in correctly as indicated in Lab 8. You need to check that your program works with the data files by testing various files. For example, blank lines at the end, should not break the program if the data is formatted as specified.

**Assignment Menu option 3** has been changed to incorporate sample *Pearson Correlation Coefficient* (**sPCC**) as shown in Lab 10.

**Assignment Menu Option 4** is updated in this lab to include the *Mean Absolute Deviation* (**mad**) calculation along with the existing standard deviation. See exercise d of Lab 9.

Menu option 4 output has the mad calculation next to the standard deviation calculation.

Average wind speed (km/h), average ambient air temperature and total solar radiation in kWh/m2 for each month of a specified year. The *standard deviation* ***stdev,*** and *Mean Absolute Deviation* ***mad*** is printed as *(****stdev****,* ***mad*** *)* next to the average. (print to a file called “*WindTempSolar.csv*”). Note that nothing else has changed in menu option 4, except the inclusion of the **mad** calculations.

**Updated output format for option 4 of assignment**

*Year*

*Month,Average Wind Speed(stdev, mad), Average Ambient Temperature(stdev, mad), Total Solar Radiation*

Example output format is: (using made up data)

*1905*

*January,5.5(1.2, 1.1),25.5(12.2, 11.3),196.4*

*February,4.5(3.1, 2.9),27.5(10.1, 10.5),200.3*

Check that none of the maths/stats routines are coupled to weather data type.

Encapsulating the std::map in your own template class Map can enable bonus marks to be given, as long as your class Map is minimal but complete. Note that the primary behaviour of the Map is to associate one type with another type. If you provide methods that are not directly relevant for this requirement, then the bonus cannot be given.

That means there is a very small number of public methods in class Map, not necessarily the same as the methods in std::map. Here is an idea to consider for the bonus:

template <class T, class U>

class Map{

public:

// only a few. Aim for no more than 4 if you can

private:

// could be the std::map

};

Bonus marks only if you have minimal but complete number of methods in Map that can defend in the documentation. Unit test carefully.

One example usage would be:

Map<std::string, int> popularity;

The assignment should build with CodeBlocks set to C++11 with all warning settings enabled. This will ensure that you are using the correct build mechanism, and the compiler is doing the hard work for you by checking and alerting you to problems.

Enable the CppCheck plugin in CodeBlocks and use that to check for issues. Download and install CppCheck (only the command line version is needed for the plugin when installing after download). <https://cppcheck.sourceforge.io/>

Then go to CodeBlocks Settings, Environment. Scroll down to CppCheck icon and point the CppCheck application to cppcheck.exe. Ignore the Vera++ tool for now. Please see diagram below.

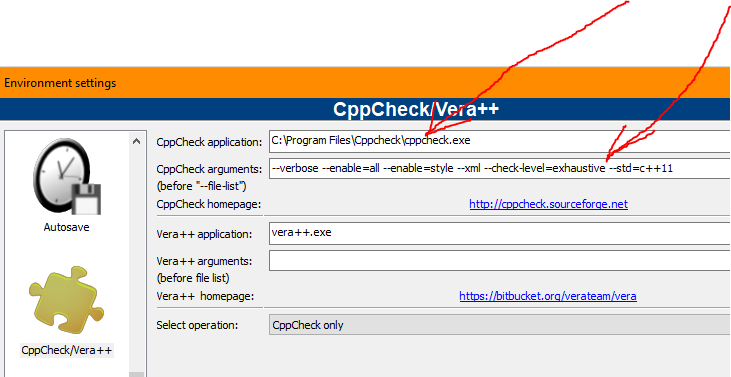


Figure showing CppCheck settings.

Once done, use the Plugins menu in CodeBlocks and run CppCheck over your code.

Consider using the Valgrind tool suite to check for memory leaks[[1]](#footnote-1) in your code. <https://valgrind.org/info/about.html>

In the demonstration/submission, you will be addressing the 3 listed objectives shown above.

**All students** demonstrate progress on your assignment/project to your tutor during class time.

You can be asked to explain your reasoning behind the approach you have taken. You can also be asked to explain why alternatives are not as good as the approach you have taken.

Note that a lab exercise from Lab 10 advised demonstration of the assignment’s progress.

The demonstration in this Lab 11 is assessed.

Ideally you would be using **private** versions of code management tools like github, gitlab or bitbucket, etc. If you are not using these, then, as a minimum, OneDrive provided to you as a student via your student account should be used. OneDrive is not an ideal solution when working with code, but better than physical drives or nothing.

**All students** would need to send a short write up (see LMS) of their progress and a zip file containing their current state of their assignment and addressing the 3 items listed in the objectives. Upload into LMS in the area marked “*Last Assessed Lab*”.

Zip up the entire assignment directory and upload into LMS. This is the version that was demonstrated to your tutor.

Please put in an *evaluation.txt* file. It should not be longer than an A4 page. The write up addresses the 3 listed objectives of this submission. The reasoning/justification (**why**) for the approach you have taken should be included in your personal evaluation. Include discussions/lessons learnt during the in-person assessment.

NOTE: The submission is not marked. It is your personal demonstration that is marked. The submission in LMS allows your mark to be recorded in LMS.

**Advice**

If you submit your assignment without your tutor knowing how you are progressing with the assignment in the weeks preceding the due date, your assignment mark may not be included in your final mark. You may also be called in for a coding interview.

1. If using linux. Something similar is Dr. Memory <https://github.com/dynamorio/drmemory> available on multiple platforms. [↑](#footnote-ref-1)