ICT283 Lab 8 exercise (not assessed in this lab but must be completed)

Objectives:

- Complete and catch up with all past practical work
- To learn about issues when doing OO re-design
- Learn to use the STL vector
- Learn the Queue data structure
- Learn uses of the stack data structure
- To do testing
- Preparation for assignment 2 - Question 4 below must be completed now as it one of the requirements for assignment 2. Assessment of this question is part of Assignment 2. **You will need to submit question 4 to access later material**.

It is very important not to fall behind with these exercises.

You should note that even though an exercise is not assessed, not attempting the exercise would make it very difficult for you to understand subsequent material. You would also have problems with assignment 2 if you do not complete the labs.

If you want to work on you own computer, install graphviz first, then install doxygen.

Exercise

Do **not** start coding until you have worked out exactly what is required. Do this on paper. Draw a UML diagram illustrating how the classes are connected and being used. Make sure you document all code using doxygen style comments.

Think about the test data you will use to demonstrate that your program works. How would you write a test plan? See Lecture notes for Topic 8. The testing needed for later units (folder "*testing in 3rd year units*") provides examples from the final project unit that all IT students must complete. For this unit, use the material from Topic 8 lecture. The samples in "*testing in 3rd year units"* will give you an idea of what will be expected of you in third year.

1. If you have fallen behind in any of the past practical work, please catch up this week.

   Convert the internal representation of your template Vector class that you had to write for your assignment 1 so that your Vector uses an STL[1] vector. Your *Vector.h* will have #include <vector>. The approach is similar to the Queue class from the lecture notes where our own Queue class encapsulates the STL queue. Topic 7 had a similar example for Stack. Work through both the Stack and Queue examples before making the change to your *Vector.h* of your Vector class.

   This is a "composition" relationship where your Vector encapsulates an STL vector and this was not permitted for assignment 1, but now you can use the STL vector when encapsulated in your own Vector. If you use this new Vector in your assignment 1, would your assignment 1 require code changes? Try it and see. If the assignment (application side – main program) requires changes in code, then your Vector is not done properly. The client program should depend only on the public interface of your Vector.

   The only changes needed should be within the body of the methods of your Vector class and the private data to make use of the STL vector. There should be no change to your client code (main program and other code) which is dependent on *your* Vector. If there are changes to the client code, then you know that your Vector was not well designed for the assignment. Take note of change issues that arose. You may want to discuss this with your tutor. Can you use your new Vector to store other objects? Run this new

[1] If you actually used the STL vector in assignment 1, then you have violated the assignment requirements. Next time read the specifications more carefully. You should also pay attention to the question and answer file.

Vector through the same unit tests used for lab 5 and assignment 1. Can your Vector be passed as references and const references to other routines and be utilised in those routines?

2. Do this exercise after you have completed all the above. Implement the calculator example shown in Lec-23 Animation.ppt. For ease of testing, read the expressions from a file which has one line per expression. If you are running short of time, do question 4 (below) first and then come back to this question.

3. Examine the Queue implementation. Is anything else needed as a method? Change the sample implementation so that no STL data structure (or algorithm) is used. Class interface must not change and the QueueTest program code must not be touched but should still build and run correctly.

4. For this exercise, you will extend assignment 1 to read multiple data files. You will continue the same Vector you used for assignment 1. You will need to be able to read multiple data files for assignment 2, so the multiple file reading problem needs to be solved now. Run assignment 1 with multiple files being used to address the menu options. The files you need for assignment 2 are already in the *data* folder.

   Use the following approach:
   a. Use data files for at least 4 different years. Keep all 4 files in a sub-directory called *data*. You do **not** need to download any more data as sufficient sample data has been provided in the *data* folder.

   b. Copy the file names of the CSV data files into a text file called *data_source.txt*. Your program will first open *data/data_source*, read each CSV data file name from *data_source.txt* and load the contents of the CSV data file into your assignment 1. You must not assume that the CSV data files are listed in any sort order in *data_source.txt*. A high level algorithm required for assignment 2 that you should implement now:

```
Open data/ data_source.txt

For each CSV file listed in data/data_source.txt
      Open the CSV file and read the data into the data structure you
        used for assignment 1
      Close the CSV data file once reading is complete
Endfor

Display Menu
```

   The program reads all files and loads the appropriate data structures before the menu is displayed to the end user.

   Menu options should now be usable for several different years.

   Add more data files to *data_source.txt* and test your Assignment 1. You must do this now before the data structure changes are incorporated for Assignment 2.

   Note:

   - You should test your program in question 4 using more data files. Question 4 and Assignment 2 needs to work with multiple data files at a time with a variety of column (field) arrangements. You do **not** need to download any more data as sufficient sample data has been provided. Download from http://wwwmet.murdoch.edu.au/ if you want to. When downloading, use the data group on the left, don't tick UTC date/time, and select the required columns of data at the web site. Do not select Rainfall (10min) and Evaporation (10min). You will get a .csv file that will have the sensor key at the top.
     - If you download your own data, open the csv file in **Microsoft Excel** and delete the top few rows but keep the header row. Check that the data/time format is correct in Excel, and

then save the file as a .csv file in the data directory for this lab. If the date is not in the *dd/mm/yyyy* format, use Excel's help to find out how to change the date format, and then save as indicated earlier. Check that the time values are sensible. You do this by checking that there should not be large values of solar radiation recorded when it is supposed to be dark (no sun) in Perth. If you see large values recorded when it is dark, the time is not WA time (WAST) but is UTC (like GMT – 8 hours behind). Open the csv file in **notepad++** (https://notepad-plus-plus.org/) to examine what your program will see when reading in the file.

- You need to complete question 4 now as you will need to submit it to access later material. The algorithm being implemented in this question is part of assignment 2.
- Application tests must be carried out. The spreadsheet gives an example.
    - Please do not fake/falsify any test result. A mark of 0 will be awarded if any fake or falsified result is found by the marker.
    - More detailed examples are found in the folder "testing in 3rd year units". This level of detail is not needed in this second-year unit.

5. Once you have question 4 above working, change the Vector in the assignment to the one used in question 1 above. Test to make sure everything works as before. Note that you shouldn't have to make changes to code other then #include of the question 1 Vector instead of the Vector from assignment 1.