ICT283 Assignment 2 2021

Objectives:

1. design and write good structured and object oriented C++ programs;
2. design and write well documented C++ programs that use programmer designed data structures;
3. demonstrate that you can use specified parts of the Standard Template Library;
4. design and execute application test plans;
5. discuss (and apply) the theory and application of data structures and the algorithms that use them and are used by them;
6. design and implement solutions that adhere to given specifications and requirements;
7. acquire further experience with design and code maintenance;
8. think critically about the problem solving approaches that you use.

You do not work in groups for this assignment as this is an individual assignment.

Worth:
35% of the unit

Due:
4 pm, Perth, WA time Last day of the semester/trimester. Submission date/time in the LMS submission area will override this.

How to submit (also see unit guide - section on Assignment/Project submission/return:
Internals: LMS
Externals: LMS

For submitting in LMS, zip up the entire folder. Make sure that you have included all needed files. Do not include temporary files or files not relevant to the assignment. After submitting the assignment, log out of LMS. Then log back in again. Check that the assignment is submitted.

Name the zip file with the unit code, Assignment number, your name, student number.
ICT283_Asignment2_Samuel_A_Bent_00700707.zip (submission by Sam A. Bent with student number 00700707)
or alternatively, ICT283_Asign2_Samuel_A_Bent_00700707.zip

Textual submissions should be type-written. External documentation can only be in the following formats:
Text (.txt)
PDF (.pdf)
RTF (.rtf)
HTML (.html)
Image formats : PNG, GIF, JPG, TIFF, BMP. (BMP and TIFF cannot be used for Web

documents)

Do not submit Word .doc or .docx files.

Assignment cover sheet requirements are listed in the unit outline found in the Admin area. LMS submissions do not require submission of a cover sheet but you should ensure that the requirements are met.

Mandatory Readings/work needed:

- Unit textbook. "*C++ Programming: Program Design Including Data Structures*" by D.S. Malik.
- Lecture notes.
- Complete lab exercises.
    - You need to complete all exercises up to laboratory 10 to cover the data structures requirements of this assignment.
    - Exercise d of Lab 10 needs to be submitted before submitting this assignment.
- Question and Answer files (QandA) if these exist in the assignment 1 and assignment 2 area. This QandA file for assignment 2 can be updated on the day of the assignment submission if someone asks a question on that day and the question and answer is deemed suitable for sharing with everyone.
- C++ Coding Standards ebook by Sutter and Alexandrescu from the library site.

Assignment Question:

This assignment continues from exercises d and e of laboratory 10. The assignment is mostly lab 10 exercise e.

Assignment 2 requires the template BST data structure from lab 10 exercise d with function pointers[1]. The STL map is also needed.

If you have not completed labs 8, 9 and 10 you will need to complete these urgently as this assignment is a continuation of those labs.

Design an object-oriented solution and implement the solution in C++ to solve the problem described below.

Historical data for solar radiation and wind speeds was obtained (along with other sensor information) from http://wwwmet.murdoch.edu.au/downloads (site is down at the moment). Weather data in the format we are after has been available from the middle of 1998 till the present day. Data is logged at intervals of 10 minutes. Sample data in comma-separated value text files is made available for this assignment. Each file contains approximately a year's worth of data for multiple sensors. Some files contain more; some contain less than a year's

[1] Lamda functions would not count towards this requirement.

worth of data. Data for each date-time recording are on separate rows. Within each row, the value for each sensor is separated by a comma. The sensor codes are the same as in assignment 1. The data directory has the required data and information on sensor codes.

To understand the nature of the data, complete lab 8. Your program must be able to read data from multiple data files. Completion of exercise 4 of lab 8 is essential for reading the data files.

You should provide a suitable menu with an exit option in your main program. When designing the output, imagine yourself as the user of the program. You want the user interaction to be user friendly on the command line. Do not use GUI interaction.

Menu Options 1 to 4 are the same as assignment 1. Multiple data files are to be used as in exercise 4 of Lab 8. Loading of files must be exactly as specified in exercise 4 of Lab 8**.**

To test your program we will be using the file data/met_index.txt and substitute the file names that you have there with our own test file names. If our test files do not get loaded or produce incorrect output, the program will be assessed as "not working". Make sure that the program is tested against manually calculated results recorded in your test plan

New menu option: – Menu option 5.

This menu option is to be attempted only when all other menu options are working to specifications. Make sure the other menu options have been checked to ensure that they are working using manually calculated results.

Proper completion of this option will give you a 10 marks bonus**.** If you attempt this option when *any* of the other options is not working, no bonus marks would be awarded.

It is theoretically possible to get 110 marks if everything is done perfectly. However the maximum mark that will awarded is capped to 100.

Given a Date in the form d/m/yyyy, show the times for the highest solar radiation for that Date. Output to screen in this format: (There can be one or more time values that are output.)

```
Date: 11/1/2019
High solar radiation for the day: 1046 W/m2

Time:
10:40
11:30
```

Add the test plan of the above menu option to your assignment 1 test plan.

Make you sure you have manually checked the expected output. This manual checking applies for all the menu options.

Menu option 6:
Exit the program.

The date, month and/or year are specified by the user. Your program asks for these on the command line and the user types in the required values and presses the "Enter" key. See instructions in assignment 1.

Menu options are selected using numbers 1 to 6.

Processing:
Your program must first load the data from multiple files. After loading the data, a menu is displayed to the user.

The required data structures (see below) must be used to provide the functionality for the menu items.

Heed the advice and lessons learned in all prior labs. Complete all readings and lab work from topics 1 to 10 and, as a minimum, exercise e of lab 10, before starting to work on this assignment. Completion of Labs 8, 9 and parts of lab 10 is particularly important for this assignment. Completing lecture material from Topic 11 (*Lect-31.ppt*) may also provide you with ideas about how to resolve some data structure issues.

A number of data files are already provided to you in the data folder. Do not move these out of the data folder.

Data Structures:
STL data structures and algorithms can be used in this assignment.

You must use the template BST from Lab 10d. You should modify the methods used in lab 9 to **make use of function pointers (as in lab 10)**. If you modify your template BST for use in this assignment, this new tree must also work in Lab 9 without any code modification other than that needed to deal with function pointers.

The template BST must not be modified in such a way that it is only relevant for this assignment. The BST should be reusable in other contexts where a template BST is useful.

You must use the STL map (or map variants).

The use of template BST with function pointers and STL map is *mandatory*, so think very hard and do not start coding until you have thought through the issues. There is no need to use both BST and map for every menu item. You can use one or the other, or combinations. Hint:

There are issues that you have to resolve before you code.

You may use other data structures together with the STL map and template BST.

You are not told how to incorporate the template BST or the STL map into the work from exercise 4 of Lab 8, so whatever approach you come up with will require written justification. *Vector is optional*. If you decide to use Vector as well, justification is needed for that too

You will need to explain the rationale to use these data structures in *your* particular way. The data structures can be used in many ways and so we want to know the thinking behind your choice of usage. Write down what you are thinking and the relevant theory you have used to guide your thinking. There may not be a perfect answer to how to combine the required data structures as we are looking at what are your thinking processes when you use the data structures. You will need to highlight issues with your chosen approach as well (pros and cons).

You should be careful that you do not have data structure classes that do I/O. You may want to have dedicated I/O classes instead or let the main program deal with I/O. Make sure you modularise your main program. Do not write monolithic methods or routines.

Class declarations and implementations should be separated. Template classes would have the implementation after the class declaration in the same header file.

Write a test plan to test the various menu options. Actual expected output has to be calculated using a spreadsheet. This expected output is to be in the test plan table.

Any advice and further clarifications to these requirements would be found in the QandA file in the assignment 2 area. Questions and Answers (if any) would also be found in this file.

Documentation: *(Printed versions apply only when there is a notification in LMS asking for hard copies. Normally hard copies would not be asked for. Ignore the advice below for printed copies if there is no notification in LMS asking for printed versions.)*

- Rationale for the use of the BST and map (also Vector, if used) data structures in the program. Provide pros and cons for your particular usage. (printed and softcopy)
- Doxygen output in html for submitted source code. It shows all information - as was done in the practice for week 2. All *.h* files should have doxygen comments as shown in the supplied *modelfile.h.* Implementation files *(.cpp*) have normal comments (soft copy)
- Application test plan for the client program ("main program"). You can use the sample table from Topic 8, *Lec-23-TestData.xls*. Make sure you actually test and mark in the test plan. Provide output of test runs. Label the test runs with the test number in the test plan.
- Executive summary indicating what works and what does not work. (printed and softcopy). The name of the summary file is ***evaluation.txt***. This file gets created using a

text editor.

Do not print code. Code will only exist as soft copy.

Minimum requirements:
You must provide all of the following;

- Documentation as listed above.
- Program that builds (code::blocks only) and runs. Build mechanism must be provided.
- Source code with doxygen style comments for declarations in .h files.
- Executable program with associated data files in a separate directory called **"*executable*"**. Make sure that the executable runs on a machine that does not have a compiler. All associated data files must be provided here too in a data sub directory.
- A declaration indicating what works and what does not work in your program. This declaration should be provided as a separate document called "***evaluation.txt***". Test plans have a lot of detail and this detail is not needed in this file. *The file* ***evaluation.txt*** *is only a summary – like an executive summary* – done as dot points.
- A declaration that your entire submission was checked for *malware*. This declaration goes at the top of the text file "***evaluation.txt***". Make sure you actually check.

Marking

Evaluation (pros/cons) and rationale for data structures.	10%
Program Design (determined from code and doxygen output) – *Full Doxygen output in html* is needed to get marks for this section.	10%
Working Program (includes C++ code, coding style, comments, C++ classes, test plan) – *Test plan* and *evaluation.txt* need to be provided to get marks for this section.	40%
BST: C++ Design, Implementation and use, including function pointers.	30%
Use of **STL map** (or map variants)	10%
Total	**100%**
Bonus Option 5 (only if options 1 to 4 work)	10%
Final mark capped to	**100%**

"Full doxygen output" means you select to output everything in html in the doxygen tool settings.

Please use the minimum requirements and documentation requirements as a check list to ensure nothing is missed. Some items asked for in assignment 1 are not submitted in assignment 2.

If unsure of anything, please ask early, preferably when completing lab 10. Please do not wait for the last few days of semester before asking.

This assignment must be submitted. Please check the unit guide for the requirements for passing the unit – section on “Determination of the final grade”.