UNIVERSITY EXAMINATIONS

OCTOBER/NOVEMBER 2023

COS3711

Advanced Programming

Welcome to the COS3711 exam.

Examiner name: Mr. S. Mhlana
Internal moderator name: Ms. P Mvelase
External moderator name: Ms. I Ngomane (University of Mpumalanga)

This paper consists of 8 pages.

Total marks: 80

Number of pages: 8
Instructions:

- You may type your answers in a word processor (and then print to PDF) or handwrite your answers (and then scan to PDF) for submission.
- This is an open-book exam. Answer all questions. Please answer questions in order of appearance.
- The mark for each question is given in brackets next to each question.
- Note that no pre-processor directives are required unless specifically asked for

Additional student instructions

1. Students must upload their answer scripts in a single PDF file (answer scripts must not be password protected or uploaded as "read only" files)
2. Incorrect file format and uncollated answer scripts will not be considered.
3. NO emailed scripts will be accepted.
4. Students are advised to preview submissions (answer scripts) to ensure legibility and that the correct answer script file has been uploaded.
5. Incorrect answer scripts and/or submissions made on unofficial examinations platforms (including the invigilator cell phone application) will not be marked and no opportunity will be granted for resubmission. Only the last answer file uploaded within the stipulated submission duration period will be marked.
6. Mark awarded for incomplete submission will be the student's final mark. No opportunity for resubmission will be granted.
7. Mark awarded for illegible scanned submission will be the student's final mark. No opportunity for resubmission will be granted.
8. Submissions will only be accepted from registered student accounts.
9. Students who have not utilised the proctoring tool will be deemed to have transgressed Unisa's examination rules and will have their marks withheld. If a student is found to have been outside the proctoring tool for a total of 10 minutes during their examination session, they will be considered to have violated Unisa's examination rules and their marks will be withheld. For examinations which use the IRIS invigilator system, IRIS must be recording throughout the duration of the examination until the submission of the examinations scripts.

10. Students have 48 hours from the date of their examination to upload their invigilator results from IRIS. Failure to do so will result in students deemed not to have utilized the proctoring tools.
11. Students suspected of dishonest conduct during the examinations will be subjected to disciplinary processes. Students may not communicate with any other person or request assistance from any other person during their examinations. Plagiarism is a violation of academic integrity and students who plagiarise, copy from published work or Artificial Intelligence Software (eg ChatGPT) or online sources (eg course material), will be in violation of the Policy on Academic Integrity and the Student Disciplinary Code and may be referred to a disciplinary hearing. Unisa has a zero tolerance for plagiarism and/or any other forms of academic dishonesty
12. Listening to audio (music) and making use of audio-to-text software is strictly prohibited during your examination session unless such usage of the software is related to a student's assistive device which has been so declared. Failure to do so will be a transgression of Unisa's examination rules and the student's marks will be withheld
13. Students are provided 30 minutes to submit their answer scripts after the official examination time. Students who experience technical challenges should report the challenges to the SCSC on 080 000 1870 or their College exam support centres (refer to the Get help during the examinations by contacting the Student Communication Service Centre [unisa.ac.za]) within 30 minutes. Queries received after 30 minutes of the official examination duration time will not be responded to. Submissions made after the official examination time will be rejected according to the examination regulations and will not be marked. Only communication received from your myLife account will be considered.
14. Non-adherence to the processes for uploading examination responses will not qualify the student for any special concessions or future assessments.
15. Queries that are beyond Unisa's control include the following: a. Personal network or service provider issues
    b. Load shedding/limited space on personal computer
    c. Crashed computer
    d. Non-functioning cameras or web cameras
    e. Using work computers that block access to the myExams site (employer firewall challenges)
    f. Unlicensed software (eg license expires during exams)

    Postgraduate students experiencing the above challenges are advised to apply for an aegrotat and submit supporting evidence within ten days of the examination session. Students will not be able to apply for an aegrotat for a third examination opportunity. Postgraduate/undergraduate students experiencing the above challenges in their second examination opportunity will have to reregister for the affected module.
16. Students suspected of dishonest conduct during the examinations will be subjected to disciplinary processes. UNISA has a zero tolerance for plagiarism and/or any other forms of academic dishonesty.
17. Students experiencing network or load shedding challenges are advised to apply together with supporting evidence for an Aegrotat within 3 days of the examination session.

There are many who make the argument that climate change is going to affect weather patterns, and, thus, the amount of rainfall that is going to fall. Some may get more, while others will get less. Tracking rainfall is, therefore, an important task.

For the sake of this scenario, you may assume the following about designing an application that tracks rainfall.

- Rain data is stored using 3 pieces of data: the station that recorded the rainfall, the date for which the rainfall is recorded, and the amount of rain in mm.
- Obviously, there must be some way of recording all this data in a container.
- There should be a way of graphing the data. Initially, only a bar graph and a column graph are required. The data that is passed to the graphing application is an XML representation of the container holding all the rainfall data.
- Clearly, then, there needs to be some way of getting an XML representation of the rain record.
- The main client will be responsible for holding the record of all the rainfall data, for getting the XML representation of all the rainfall data, and for passing this to the user's choice of graph.
- Appropriate design patterns should be used as necessary.

Question 1 **[25 marks]**

1.1 Considering the scenario given above, draw a partial UML class diagram that captures the scenario. You should include the necessary classes, class attributes, and class relationships that are mentioned in the scenario. Class constructors and member access specifiers are not required. However, you should ensure that you include all the data members and member functions that show how data will be moved around the application. You should include the Client/GUI class.
[You may use a software tool to create the UML class diagram. Use underlining to represent italics in hand-drawn UML class diagrams.] (20)

1.2 It has been argued that the design pattern that would be used here is a behavioural pattern. Do you agree, indicate which pattern would be used and why it is or is not a behavioural pattern? (2)

1.3 It was decided to provide the class that holds the rain data with some reflective functionality. That is, a `getData()` function should return a single data string that contains all the data held by the class via the class's meta-object, in a property named `data`. This function is used only by the meta-object and should not be available to users of the class.

Write the class definition for this class showing how this would be set up. You are not required to include functionality or data members not mentioned above, or to code the implementation of the `getData()` function. (3)

Question 2 **[30 marks]**

Consider now the generation of the XML text that will be passed to the graphing applications.

The format for the XML text to be generated is the following.

```
<rainRecord>
    <rain date="2023/01/01">
        <station>AagA100</station>
        <mm>10</mm>
    </rain>
    <rain date="2023/01/02">
        <station>BcdB123</station>
        <mm>5</mm>
    </rain>
</rainRecord>
```

2.1 If the date is stored as a `QDate` (say, `QDate date`), give the code you would use to convert this data into the string format that is used in the XML text above? (2)

2.2 Suppose that a class named `RainXml` would be used to generate the XML text for all the rain data.

```
class RainXml
{
public:
    RainXml();
    RainXml getInstance();
    QString writeToXml(/*passing rain data*/);
private:
    RainXml instance;
    bool checkStationCode(QString stn) const;
    QRegularExpression re;
};
```

This class is supposed to be implemented using a singleton design pattern.

2.2.1 The class definition code provided above does not correctly implement the classic singleton design pattern. Correct it so that it does. You are not expected to indicate the container that is used to pass rain data. (3)

2.2.2 Would you agree with the decision to make this class use the singleton design pattern? Explain your reason clearly and persuasively; note that marks are only allocated for the reasoning. (2)

2.3 The `checkStationCode(QString stn)` function from the class definition in 2.2 is used to ensure that the station code meets the correct format required, where the `QString` parameter is the station code that needs checking. The returned Boolean is `true` if the code meets requirements and `false` otherwise.

```
bool RainXml::checkStationCode(QString stn) const
{
    // add code here
}
```

The station code should meet the following requirements.

- Should begin with a capital letter.
- This is followed by any 2 lowercase alphabetic characters.
- This is followed by the same capital letter as the initial character of the code.
- The numeric part of the code is made up of any 3 digits, where the first digit cannot be zero.
- There should be no other characters before or after this code.

2.3.1 Write the `QRegularExpression` that would be used to check codes for correctness.

```
QRegularExpression re(/*what would you put here*/);
```

(7)

2.3.2 Which anti-pattern would be involved if the code were not checked for meeting requirements? (1)

2.3.3 Provide the code for the `checkStationCode(QString stn)` function assuming that `QRegularExpression re`, defined in 2.3.1, is a data member of the `RainXml` class. (2)

2.4 The following code stub is used to generate the XML text given above. Using the required XML format from the start of Question 2, the `data` property from the meta-object from 1.3, and the `checkStationCode()` function from 2.3, complete the code where indicated by comments.

Assume that each item of rain data can be obtained from the meta-object property set up in 1.3 – it returns the data in the form: `StationCode:yyyy/mm/dd:mm`. For example, if 10 mm of rain were received at station AagA100 on 1 January 2023, the data would be in the form `AagA100:20230101:10`.

You are not required to provide the code for the `writeToXml()` function parameter or the main loop that loops through all the rain records. You may assume that for each pass through the loop, you have access to a pointer `r` that points to a rain record.

```
QString RainXml::writeToXml(/*passing rain data*/)
{
    QString xmlOutput;
    QXmlStreamWriter writer(&xmlOutput);

    // do initial setup of xml text

    // loop through each rain pointer named r (do not code this)
    {
        // use the meta-object to get the required data

        //if the station code passes the test
        {
            // set up the <rain> tag and its sub-tags as required
        }
    }

    // end xml text

    return xmlOutput;
}
```

(13)

Question 3 **[25 marks]**

The plan is to search the whole record of all rainfall data for a particular station's data, and then present this data on the client screen. This search should be done in a thread.

Consider the class implementation stub below (where `stn` is the station's data that is required).

```
StationThread::StationThread(/*all data*/, QString stn)
    :  record{/*all data*/}, station{stn}
{}

void StationThread::doSearch(){
    foreach(/*rain record in the data*/){
        //get the station, date, and mm as strings
        if (/*this station in the data*/ == station)
           emit  foundStation(/*date  as  string*/,  /*mm  as
string*/);
    }
}
```

3.1 Write the class definition for the `StationThread` class, remembering that it should be run as a thread. (8)

3.2 Consider the code below that is run when data for a particular rain station is found by the code running in the thread, where `QTableWidget *tableWidget` and `int row` have already been declared and appropriately initialised.

```
void Client::handleFound(QString date, QString mm){
    QTableWidgetItem                          *dateItem{new
QTableWidgetItem(date)};
    QTableWidgetItem *mmItem{new QTableWidgetItem(mm)};

    tableWidget->setItem(row, 0, dateItem);
    tableWidget->setItem(row++, 1, mmItem);
}
```

Assume the following code in `Client`.

```
QThread *t{new QThread};
StationThread       *st{new       StationThread(/*passing
parameters*/)};
```

Write the code that would follow these declarations to get the thread running with the `StationThread` object, ensuring that the data is received from the running thread and passed on to the `handleFound()` function. (6)

3.3 Would you agree that the `QTableWidget` used in 3.2 is the best approach that can be used for displaying a station's rainfall data on the client window? Give reasons for your answer. Note that marks are only allocated to your reasoning. (2)

3.4 As a way of managing a backup/restore functionality, it has been decided to subclass the `QTableWidget` class to implement a classic memento design pattern. The start of the class definition is provided below.

```
class MyTableWidget: public QTableWidget
{
private:
    friend class MyTableWidgetMemento;
    MyTableWidget();
    MyTableWidgetMemento* createMemento();
    void setMemento(MyTableWidgetMemento *m);
};
```

3.4.1 What components usually make up the classic memento pattern, and which classes in this scenario would represent those components? (3)

3.4.2 Explain why this implementation of the classic memento patten is correct or not. Marks are only awarded for the explanation. (2)

3.5 It has been argued that cloud computing is the best way to go when it comes to managing all the rainfall data. Explain why this would be so in terms of cost and scale. (4)