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 9 pages.

Total marks: 80

Number of pages: 9

<u>Instructions:</u>

- 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.

A large store currently sells only books and magazines, although this may well be expanded to other items at some future point. While both books and magazines have a name, we also want to know the name of the book publisher and the magazine frequency.

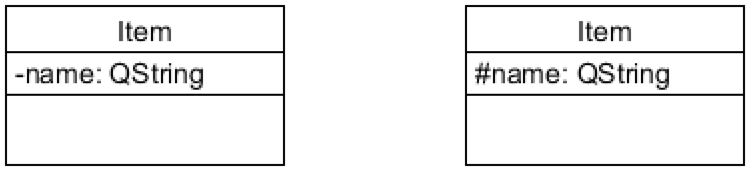
A master list of all books and items should be kept, which should have all the functionality of a QList. The client then contains an instance of this list, which can be accessed by several functions in the client.

Question 1 **[30 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.] (12)

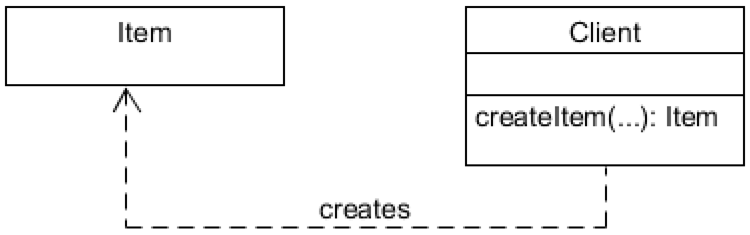
1.2 Consider the two partial UML classes depicted below.



Explain what the difference is between them and the effect this will have on the rest of the application when dealing with books and magazines. (3)

1.3 Should the item class be abstract? Note that marks are only allocated for explaining your reasoning. (3)

1.4 It was decided to use a factory method pattern when creating items for the store list. The following UML class diagram was provided.



If this representation is correct, explain in detail how the creation of items is implemented (filling in details that may be missing from the UML class diagram provided). If this representation is not correct, redraw the UML class diagram. (4)

1.5 The plan is to include an item code for each item that adheres to the following format.

- Starts with either Bk (for a book) or Mg (for a magazine).
- A number in the range 000-399 (that takes up 3 positions in the code).
- An underscore.
- One or two lowercase alphabetic characters, where the final character should not be a vowel.
- There should be no other characters before or after this code.

A QRegularExpression will be used to ensure that a string matches the required format.

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

1.5.1 Write down what you would provide in the comment in the code above. (7)

1.5.2 Which would be better to use to validate the code when a user enters it on the GUI – a regular expression or an input mask? Use an example from the required code format to show why you say so. The mark is only awarded for the explanation. (1)

Question 2 **[25 marks]**

Consider the individual and corporate customers that this store serves.

```
class Customer{
public:
  Customer();
  Customer(QString n);
  ~Customer();
  QString listToXml() const;     // returns list in XML format
  void listFromXml(QString s);   // builds list from XML format
  void setName(QString n);       // name setter
  QString getName() const;       // name getter
  QList<Customer*> *getList();    // returns the whole list
private:
  QString name;
  QList<Customer*> *list;        // list holding individual and
                                 // corporate customer instances
};
class Individual: public Customer{
public:
  Individual(QString n, QString id);
  QString getIdNumber() const;
private:
  QString idNumber;
};
class Corporate: public Customer{
public:
  Corporate(QString n, QString reg);
  QString getRegNumber() const;
private:
  QString regNumber;
};
```

2.1 Comment on the structure and content of the Customer class, providing reasons for your assessment. (2)

2.2 The Customer class represents an implementation of the serialiser pattern. Explain in detail why you do, or do not, agree. (2)

2.3 The list data member contains pointers to Customer objects. Explain why this is necessary. (2)

2.4 Write the implementation code for the parameterised Customer constructor and destructor, following good OOP programming practice. (4)

2.5 Assuming that the XML string is in the following format,

```
<customers>

   <customer type="Corporate">

      <name>Customer1name</name>

      <regNumber>A4423</regNumber>

   </customer>

   <customer type="Individual">

      <name>Customer2name</name>

      <idNumber>111111 1111 111</idNumber>

   </customer>

</customers>
```

Consider the code below for the listFromXml() function and

- where there are errors in the code, correct them;
- add code where indicated by comments; and
- add the customer to the list in the appropriate place in the function.

You may assume that the list will be empty and that there will be only two types of customers, but you cannot assume that the tags for a customer will always have the name tag first.

[If you are using a word processor, you may copy the code into your answer document and make the necessary changes.]

```
void Customer::listFromXml(QString s) {

  QDomDocument doc;

  if (!doc.setContent(&xmlString)){

    QDomElement root = doc.rootElement();

    if(root.tagName() == "customer")      {

      QDomNode node = root.firstChild();

      while(!node.isNull()){

        QDomElement element = node.element();

        if(!element.isNull()){
                if(element.tagName() == "customer type"){
                  QString type{element.attribute()};
                  QString name;
                  QString other;
                  QDomNode n = node.firstChild();
                  while (!n.isNull()){
                    // get the tag values for this customer
                    n = n.nextTag();
                  }
                }
              }
              node = node.nextSibling();
            }
          }
        }
      }
```
(15)

Question 3 **[18 marks]**

The customer list, which is implemented as a list of Customer pointers (as in Question 2), is to be displayed on the GUI using a model/view approach. The following model is required.

QStandardItemModel *model{new QStandardItemModel};

The interface should look as follows.

Type	Name	ID number	Registration number
Individual	Customer2name	111111 1111 111	
Corporate	Customer1name		A4423

3.1 As a pointer has been used for model above, describe how the allocated memory will be freed, and where the responsibility for this lies, indicating why this is so. (3)

3.2 Using an appropriate view for this model, declare the view and ensure that the model data will be visible in the view. (2)

3.3 The Customer class would have to be reflective to allow code to determine what type of instance the Customer pointer in the customer list is pointing to. Rewrite the Customer class so that it would allow for such reflective programming approaches to be used. You need not include all the functions and data members in rewriting the class that are not part of implementing reflection. (1)

3.4 Using the code stub below, expand the code so that it would add each Customer item to the model.

```
Customer customer;

QList<Customer*> *list{customer.getList()};

foreach (/*customer*/){

  // add the customer to the model

}
```
(12)

Question 4 **[7 marks]**

4.1 Each sale made in the store is recorded as a Transaction.

```
class Transaction{

public:

  Transaction(QString c, QDate d, QList<Item> i);

  /*some type*/ getBackup() const;  // see below

private:

  QString customer;  // name of customer

  QDate date;        // date of transaction

  QList<Item> items; // list of items (books/magazines) purchased

};
```

When the client application creates a Transaction, it also requests a backup of that instance to hold in memory.

```
Transaction t(/* parameters passed to construct object */);

/*some type*/ *backup{t.getBackup()};
```

Consider the implementation of this backup functionality, where an appropriate design pattern would be used to ensure that the data in the backup is not accessible by the client. You do not need to consider restoring from the backup.

4.1.1 Write the class header for the class that would support this backup functionality. (4)

4.1.2 Indicate what type would be used in the code fragment above (/*some type*/). (1)

4.2 Cloud computing takes on a shared-responsibility model. What two features are the responsibility of the cloud provider? (2)