UNIVERSITY EXAMINATIONS

UNIVERSITEITSEKSAMENS

UNISA university of south africa

COS2614
RCO2614
SECOND PAPER

May/June 2017

PROGRAMMING: CONTEMPORARY CONCEPTS

Duration : 2 Hours

75 Marks

EXAMINERS :

FIRST : MS A THOMAS

SECOND : MR CL PILKINGTON

Closed book examination.

This examination question paper remains the property of the University of South Africa and may not be removed from the examination venue.

INSTRUCTIONS:

1. This examination paper consists of 6 pages.
2. This is a closed book examination.
3. The mark for each question is given in brackets next to the question.
4. Answer all five questions in your answer book.
5. All rough work must be done in your answer book.
6. Please answer the questions in order of appearance.

GOOD LUCK!

Question 1 **[10]**

Refer to the code below:

```
#include <QApplication>
#include <QInputDialog>
#include <QMessageBox>

int main(int argc, char * argv[]){
    QApplication app(argc, argv);

    //include lines of code here

    return EXIT_SUCCESS;
}
```

Complete the above program so that

- it reads the weight and the height of a person entered in a space separated format on a `QInputDialog`. Assume that the weight and height will only be entered in kilograms and meters respectively. An example of an input in the `QInputDialog` is `50 1.58` where `50` and `1.58` refer to the weight and the height respectively of a person.
- it computes the body mass index (BMI) using the formula BMI = weight/(height×height).
- it displays the computed BMI on a `QMessageBox`.

Note:
- You can expect the input in the correct format, so no input validation is required.
- `toDouble()` is a `QString` member function that converts a `QString` to double, if such a conversion is possible.

Question 2 **[10]**

Refer to the following program to answer the questions that follow:

```
int main(int argc, char* argv[]) {
  QApplication app(argc, argv);
  QWidget window;
  QLabel* label = new QLabel("Please enter some text");
  QTextEdit* textEdit = new QTextEdit;
  QVBoxLayout* layout = new QVBoxLayout;
  layout->addWidget(label);
  layout->addWidget(textEdit);
  window.setLayout(layout);
  window.show();
  return app.exec();
}
```

The given program makes use of `QObject`'s child management facility hence, it does not create memory leaks.

2.1 Indicate, using a simple diagram, the parent objects and their respective child objects in the above program that make use of `QObject`'s child management facility. (3)

2.2 Qt provides a child management facility through the `QObject` class. Where is the `QObject` in the above program? Explain your answer. (2)

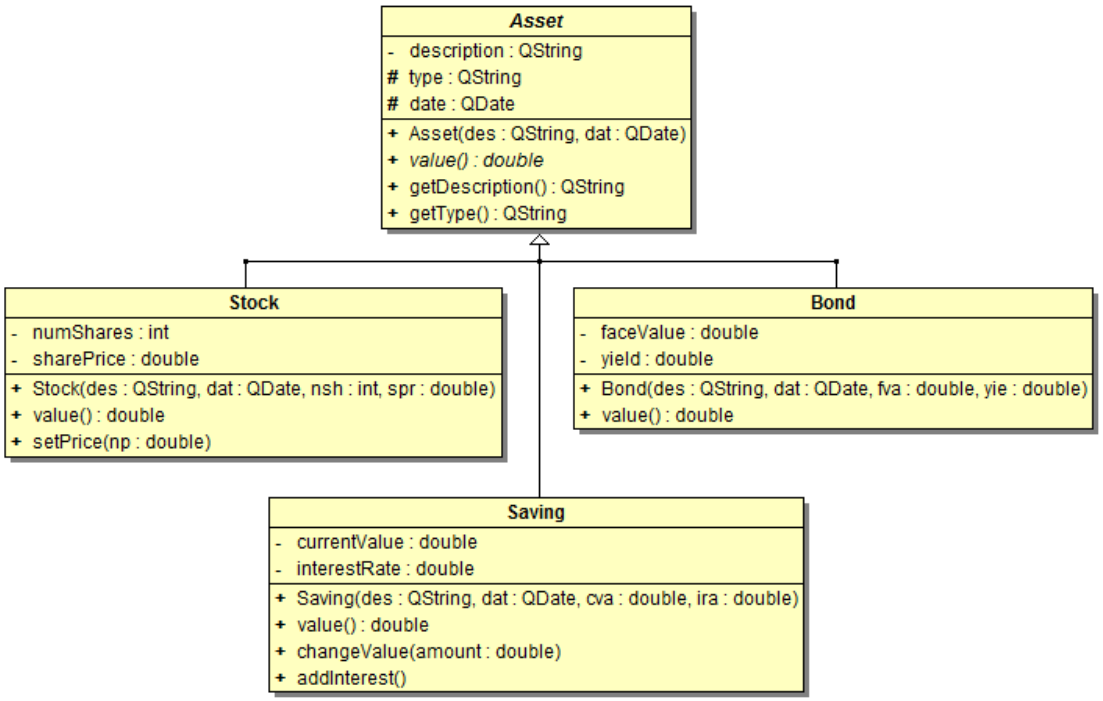
2.3 The `QObject` class provides the function `setParent(QObject* parent)` to specify a `QObject` to be its parent. Why is this function not used in this program? (2)

2.4 The program contains both stack and heap objects. Explain how the parent-child facility works when the parent is allocated on the stack and the child objects are allocated on the heap. (2)

2.5 Which design pattern is implemented in the `QObject` class to realize the child management facility? (1)

Question 3 **[20]**

Refer to the following UML class diagram to answer the questions that follow:

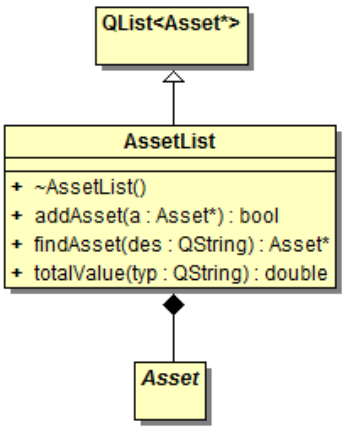


3.1 Write down the definition of the function `value()` in `Asset`. (2)

3.2 A `toString()` function returns the state of an object in a readable string format. Write down the code needed to add to each class definition a `toString()` member function for these classes so that polymorphism can be applied. No implementation is required. (3)

3.3 Write code to implement `toString()` in `Stock` so that it returns all the information about a `Stock` object, including labels for each data member. The function must use partial overriding. (4)

Consider the `AssetList` class depicted in the following UML diagram (The `Asset` class is as specified at the beginning of Question 3):



3.4 Write code for a function called `outputAll()` to output the information of all the assets in an `AssetList` instance to the console (i.e. to the standard output device). You can assume that a `QTextStream` called `cout` has been declared as a global variable. Note that the function should not be a member function of the `AssetList` class. It should be a stand-alone function that takes an `AssetList` as parameter. (3)

3.5 Explain the difference between polymorphism and polymorphic assignment, and state where they are applied in your answer to Question 3.4. (5)

3.6 Explain the concept of implicit sharing applicable to Qt containers. (3)

Question 4 **[20]**

Consider the following code and answer the questions that follow:

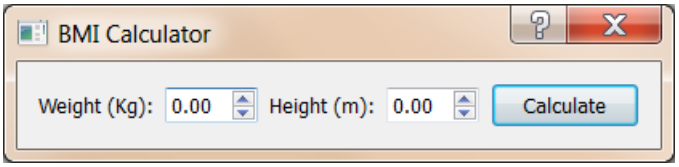
```
class BMIDialog : public QDialog{
    Q_OBJECT
public:
    BMIDialog();
private slots:
    void calculateBMI();
private:
    QDoubleSpinBox* weight;
    QDoubleSpinBox* height;
};
BMIDialog::BMIDialog(){

    QLabel *weightLabel = new QLabel("Weight (Kg):", this);
    weight = new QDoubleSpinBox(this);
    QLabel *heightLabel = new QLabel("Height (m):", this);
    height = new QDoubleSpinBox(this);
    QPushButton *calculate= new QPushButton("Calculate", this);

    this->setWindowTitle("BMI Calculator");
    connect(calculate, SIGNAL(clicked()), this, SLOT(calculateBMI()));

}
```

4.1 Complete the constructor of `BMIDialog` so that it will produce the following interface:



You do not need to rewrite all the given code; only indicate what additional code needs to be added and where. (5)

4.2 Complete the implementation of the slot `calculateBMI()` so that the weight and height entered are used to calculate BMI (weight/(height×height)). The calculated BMI is then displayed on a `QMessageBox`. (5)

4.3 Explain the principle of separating model and view. (2)

4.4 How should the given `BMIDialog` class be restructured to apply the principle of separating model and view? (2)

4.5 What is the benefit of separating model and view? (2)

4.6 Explain the purpose of signals and slots in Qt. (2)

4.7 State one similarity and one difference between signals and slots in Qt with respect to their definitions and implementations. (2)

Question 5 **[15]**

5.1 Explain two similarities and three differences between `QMap` and `QList`. (5)

5.2 Write code to create a `QMap` object with `QString` as keys and `Foo*` as values. (1)

5.3 Consider the code of the `MyMainWindow` class:

```
class MyMainWindow : public QMainWindow {
    Q_OBJECT

public:
    MyMainWindow();

private slots:
    void handleAction(QAction * a);

};
```

```
MyMainWindow::MyMainWindow() {

    setWindowTitle("My Editor");

    QAction * addAction = new QAction("&Open", this);
    QAction * changeAction = new QAction("&Change", this);
    QAction * removeAction = new QAction("&Remove", this);

    QToolBar * toolBar = new QToolBar("Main");
    addToolBar(Qt::TopToolBarArea, toolBar);
}
```

Write lines of code in the constructor to:

(a) Create a `QActionGroup` object to which the given three `QActions` are added. (3)

(b) Create a menu titled `Edit` with three items `Open`, `Change` and `Remove`. Use the `QActionGroup` in object **(a)** to add the menu items. Add the `Edit` menu to the main window. (3)

(c) Populate the toolbar object with control buttons `Open`, `Change` and `Remove`. Use the `QActionGroup` in object **(a)** to add the toolbar buttons. (1)

(d) Have a connect statement so that actions are handled by the slot `handleAction()`. (2)