HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

21.09.2022

Klausur

MMI und GUI-Programmierung

Sommersemester 2022

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer

Bewertung

Frage	Punkte	Erreicht
1	5	
2	5	
3	10	
4	15	

Frage	Punkte	Erreicht
5	14	
6	38	
7	13	
Gesamt:	100	

Ablauf der Prüfung

- **Zeit für die Lösung dieses Aufgabenblatts**: 120 Minuten.
- **Erlaubte Hilfsmittel**: Cheat Sheet am Ende der Klausur. Handgeschriebener, ein- oder beidseitig beschriebener Notizzettel im DIN-A4-Format.
- **Nicht erlaubte Hilfsmittel**: Mobiltelefon, Kommunikation (E-Mail, Web, SMS, etc.). Die Verwendung eines nicht erlaubten Hilfsmittels wird als Täuschungsversuch gewertet.
- **Stift und Papier**: Schreiben Sie ausschließlich auf dem Papier der Aufgabenblätter. Sollten die Zwischenräume nicht ausreichen, können Sie zusätzliches Papier von der Aufsicht erhalten. Beschriften Sie dieses sofort mit ihrem Namen und Ihrer Matrikelnummer. Eigenes Papier ist nicht zulässig. Schreiben Sie ausschließlich mit dokumentenechten schwarzen oder blauen Stiften.
- **Vollständigkeit des Aufgabenblatts**: Die Klausur besteht aus den Aufgabenseiten 5 – 20 und einer Funktionsreferenz auf Seite R1. Die Aufgabenblätter sind doppelseitig bedruckt. Prüfen Sie, ob alle Blätter vorhanden sind.
- **Fragen**: Melden Sie sich, wenn Sie eine Frage haben. Die Aufsicht kommt zu Ihnen, verlassen Sie nicht unaufgefordert Ihren Platz.
- **Zwischenschritte**: Geben Sie Zwischenschritte an. So können Sie selbst bei falschem Endergebnis einen Teil der Punkte erreichen.

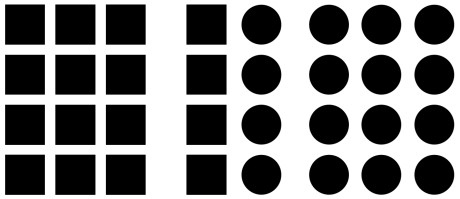
MMI

1. Erläutern Sie die Ziele von guter Usability. (5)

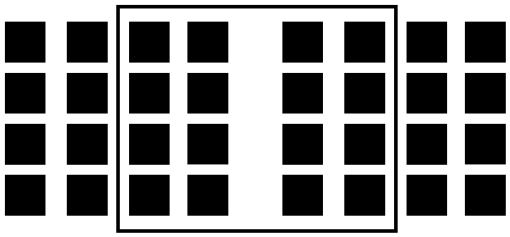
2. Erläutern Sie den Zweck eines *Wireframes*. (5)

3. Betrachten Sie die folgenden Abbildungen. Welche zwei Gestalt-Gesetze stehen hier in Konflikt? Begründen Sie kurz Ihre Antwort.

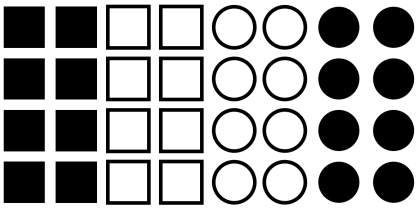
(a) (2½)



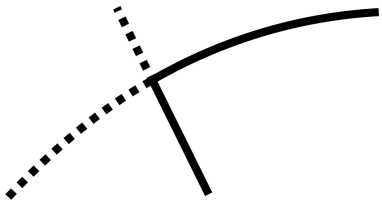
(b) (2½)



(c) (2½)



(d) (2½)



C++

4. Betrachten Sie die C++-Klasse `Stack`:

```
class Stack {
public:
    Stack();

    void push(int element);
    void pop();
    int top() const;
    size_t size();

private:
    std::stack<int> values;
};
```

Die Klasse stellt einen einfachen Stack ("Stapelspeicher") dar und bietet die folgenden Funktionen:

- `push(element)` legt das Element `element` auf den Stack.
- `pop()` entfernt das oberste Element vom Stack.
- `top()` liefert das oberste Element des Stacks, ohne es zu entfernen.
- `size()` liefert die Anzahl der Elemente, die auf Stack liegen.

Implementieren Sie die Operatoren auf der folgenden Seite.

(a) Der operator `<<` legt ein Element auf den Stack. Beispiel: (5)

```
Stack stack;
stack << 5;
stack << 2;
stack << 1;
// Stack enthält jetzt: [1, 2, 5]
```

(b) Der operator `!` löscht das oberste Element des Stacks und gibt es zurück. Beispiel: (5)

```
Stack stack;
stack << 5;
stack << 2;
stack << 1;
// Stack enthält jetzt: [1, 2, 5]
int topElement = !stack;
// Stack enthält jetzt: [2, 5], topElement ist 1
```

(c) Der operator $\sim$ leert den Stack. Beispiel: (5)

```
Stack stack;
stack << 5;
stack << 2;
stack << 1;  // Stack enthält jetzt: [1, 2, 5]
~stack;      // Stack ist jetzt leer
```

5. Schreiben Sie eine generische Funktion `getAverage` mit Hilfe eines Templates. Die Funktion erhält ein Array mit Zahlen des generischen Typs `T` und die Anzahl der Array-Elemente. Sie gibt den Durchschnitt[1] alle Array-Elemente zurück.

   Das folgende Hauptprogramm nutzt die gesuchte Funktion:

```
int main() {
    int intArray1[] = {2, 1, 5, 2};
    int intAverage1 = getAverage(intArray1, 4);
    printf("int average 1 is %d\n", intAverage1);

    int intArray2[] = {1, 2, 3, 6};
    int intAverage2 = getAverage(intArray2, 4);
    printf("int average 2 is %d\n", intAverage2);

    double doubleArray[] = {2, 1, 5, 2, 5, 1.5};
    double doubleAverage = getAverage(doubleArray, 6);
    printf("double average is %f\n", doubleAverage);

    return 0;
}
```

   Die erwartete Ausgabe des Programms ist:

```
int average 1 is 2
int average 2 is 3
double average is 2.750000
```

(Fortsetzung auf der nächsten Seite)

[1] Durchschnitt = Summe dividiert durch Anzahl der Elemente

(a) Implementieren Sie die generische Funktion `getAverage`. (10)

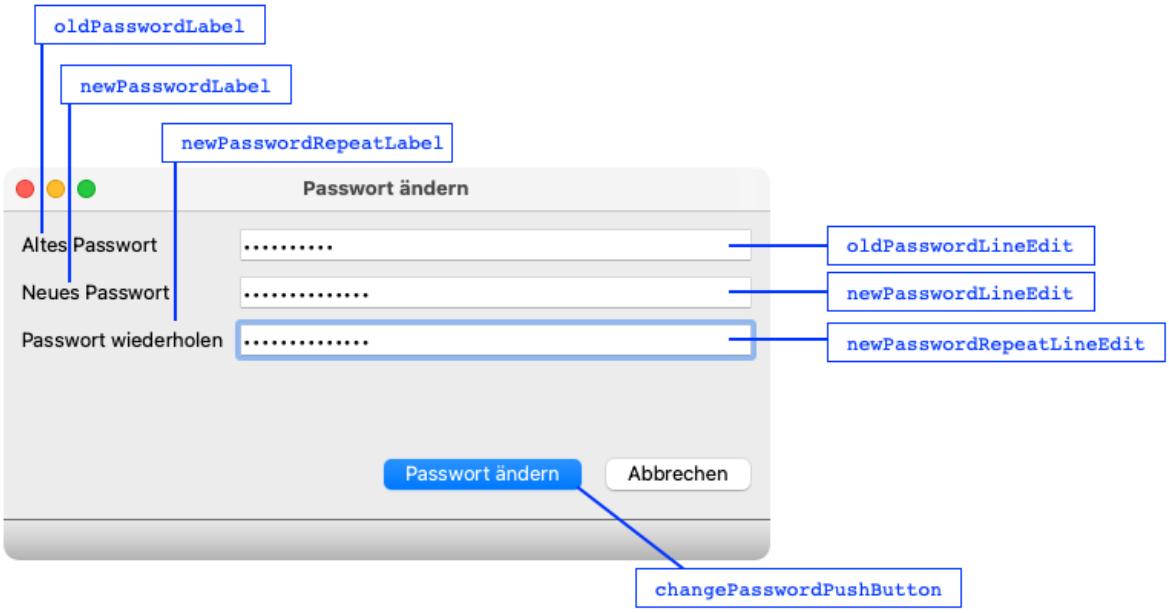
(b) Nehmen wir an, die generische Implementierung von `getAverage` und das oben gezeigte Hauptprogramm werden gemeinsam kompiliert. (4)

Wie oft kommt die Funktion `getAverage` in der Objektdatei (= im Kompilat) vor? Begründen Sie Ihre Antwort.

Qt

6. In dieser Aufgabe geht es um die folgende GUI, mit der Nutzer ihr Passwort ändern können:

Die folgende Abbildung zeigt die Namen der einzelnen UI-Elemente und demonstriert, wie sich die Ausrichtung der Elemente anpasst, wenn die Fenstergröße geändert wird:



Die GUI bietet das folgende dynamische Verhalten:

1. Die Schaltfläche “Passwort ändern” ist nur genau dann aktiviert (klickbar), wenn alle Eingabefelder mindestens ein Zeichen enthalten.
2. Wird auf “Passwort ändern” geklickt, gilt folgendes:
   - Weicht der Inhalt der Eingabefelder `newPasswordLineEdit` und `newPasswordRepeatLineEdit` voneinander ab, erscheint eine Fehlermeldung
   - Ansonsten wird das neue Password durch Aufruf der globalen Funktion `savePasswordInDatabase(QString newPassword)` gespeichert.

(a) Skizzieren Sie, wie die GUI über Layout-Manager organisiert ist, damit sie wie gezeigt auf Größenänderungen reagiert. Alle in der Abbildung genannten **IDs** der GUI-Elemente, die **konkreten Layout-Manager** und ggfs. **Spacer** sollten dabei erwähnt werden. Tipp: Die Liste der Layout-Manager finden Sie in der Referenz. (10)

(b) Die Hauptfensterklasse heißt `MainWindow`. Den Inhalt der entsprechenden Header-Datei `mainwindow.h` finden Sie unten. (8)

Machen Sie sich zunächst Gedanken, wie das oben beschriebene dynamische Verhalten umgesetzt werden kann. Ergänzen Sie in der Header-Datei dann die Slot-Deklarationen, die Sie benötigen, um das beschriebene dynamische Verhalten umzusetzen.

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
```

```

};
#endif // MAINWINDOW_H
```

(c) Implementieren Sie nun den Konstruktor der Klasse `MainWindow`. Der Konstruktor sollte den Initialzustand der GUI-Elemente setzen (sofern notwendig) und die Signal-und-Slot-Verbindungen herstellen. (10)

Tipp: Wie üblich können Sie über `ui` auf die GUI-Elemente zugreifen.

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
```

```
}
```

(d) Implementieren Sie nun die verbleibenden Slots, die Sie in der Header-Datei definiert haben. (10)

7. Ergänzen Sie die folgenden Funktionen so, dass die angegebenen Anforderungen an die Thread-Sicherheit erfüllt ist. Hierbei sollten Sie entsprechende Qt-Klassen verwenden.

   (a) `executeServerRequest` darf immer nur ein Thread ausführen (keine gleichzeitige Ausführung erlaubt). (4)

```
void callExtensiveFunction() {

    executeServerRequest();

}
```

   (b) `executeMagicServerRequest` dürfen maximal fünf Threads gleichzeitig ausführen. (4)

```
void makeMagicStuff() {

    executeMagicServerRequest();

}
```

(c) Auf `executeReading` dürfen beliebig viele Threads gleichzeitig zugreifen, es sei denn, ein Thread ruft gerade `executeWriting` auf. (5)

```
void readDaStuff() {



    executeReading();



}
void writeDaStuff() {



    executeWriting();



}
```

Qt Cheat Sheet

Signale

QPushButton / QAbstractButton

```
void    clicked(bool checked = false)
void    toggled(bool checked)
void    pressed()
void    released()
```

QLineEdit

```
void    textChanged(const QString &text)
void    editingFinished()
void    inputRejected()
void    returnPressed()
void    selectionChanged()
```

Methoden

QLineEdit

Text auslesen
QString text() const

Text setzen
void setText(const QString &)

QAbstractButton

Auslesen: Ist Checkbox angehakt?
bool isChecked()

QString

Länge ermitteln
int size() const

Slots

QWidget

```
bool    close()
void    hide()
void    setDisabled(bool disable)
void    setEnabled(bool)
void    setFocus()
void    setHidden(bool hidden)
void    setVisible(bool visible)
void    setWindowModified(bool)
void    setWindowTitle(const QString &)
void    show()
void    update()
```

QAbstractButton

```
void    animateClick(int msec = 100)
void    click()
void    setChecked(bool)
void    setIconSize(const QSize &size)
void    toggle()
```

QPushButton *(erbt von QAbstractButton)*

```
void    showMenu()
```

QLineEdit

```
void    clear()
void    copy() const
void    cut()
void    paste()
void    redo()
void    selectAll()
void    setText(const QString &)
void    undo()
```

Layout-Manager

QHBoxLayout
QVBoxLayout
QFormLayout
QBoxLayout
QGridLayout
QStackedLayout

Thread-Synchronisation

QMutex
QReadWriteLock
QSemaphore

Message-Boxen

```
QMessageBox::critical(QWidget *parent, const QString &title, const QString &text,
    StandardButtons buttons = Ok, StandardButton defaultButton = NoButton);
```