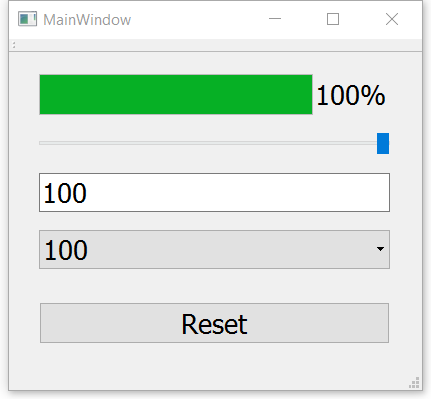
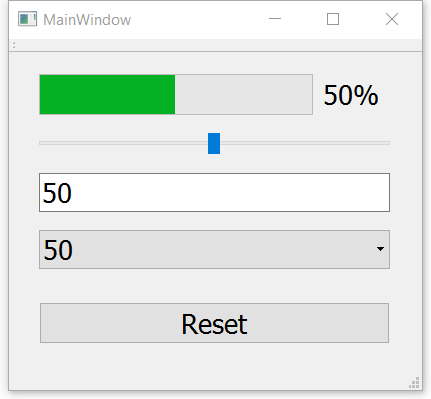
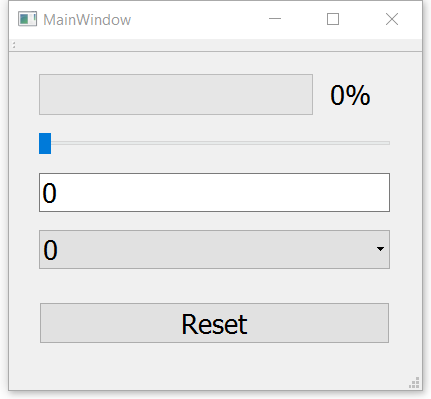
Homework 17

For this homework, fill in this word file with your answers. I suggest you to work on this problem purely on paper, with your computer and textbook closed, because a problem of this style will be given in Exam 2. The images shown below are screen shots of a simple application. The application starts with the main window that contains the following objects: a progress bar, a horizontal slider, a text input, a combo box and a push button. The progress bar ranges from 0 to 100. The slider range is from 0 to 100 and its default location is the 0th position. The text input shows the value of slider position, it can also be used to change the slider position, progress bar and combo box values. The combo box contains three options; 0, 50, and 100. Changes in combo box selections are immediately reflected in all other objects. The reset button resets the values of all objects to zero.

**(1)** Complete the code given below to implement the functionality described above. For your convenience, parts of the code are already given to you while some functions have been removed and replaced with empty boxes that you need to fill. Here is a list of QT signals for your reference: textChanged(QString), currentTextChanged(QString), valueChanged(QString), textEdited(QString), returnPressed(), editingFinished(), clicked(bool), clicked(), pressed(), released(), stateChanged(int)



**Mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

**namespace** Ui {

**class** MainWindow;  }

**class** MainWindow : **public** QMainWindow

{  Q\_OBJECT

**public**:

**explicit** MainWindow(QWidget \*parent = 0);

    ~MainWindow();

**public** slots:

**void** actByYourChange(QObject\*);

**private**:

    Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**Mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "myslider.h"

#include "mylineedit.h"

#include "mycombobox.h"

MainWindow::MainWindow(QWidget \*parent) :

    QMainWindow(parent),

    ui(**new** Ui::MainWindow)

{

    ui->setupUi(**this**);

    ui->horizontalSlider->setMinimum(0);

    ui->horizontalSlider->setMaximum(100);

    ui->horizontalSlider->setValue(0);

    ui->progressBar->setMinimum(0);

    ui->progressBar->setMaximum(100);

    ui->progressBar->setValue(0);

    ui->lineEdit->setText("0");

connect(ui->comboBox,SIGNAL(currentTextChanged(QString)),ui->comboBox,SLOT(myCurrentValueChanged(QString))); connect(ui->comboBox,SIGNAL(iChanged(QObject\*)),this,SLOT(actByYourChange(QObject\*)));

connect(ui->lineEdit,SIGNAL(editingFinished()),ui->lineEdit,SLOT(myEditingFinished())); connect(ui->lineEdit,SIGNAL(iChanged(QObject\*)),this,SLOT(actByYourChange(QObject\*)));

connect(ui->horizontalSlider,SIGNAL(valueChanged(int)),ui->horizontalSlider,SLOT(setslider(int))); connect(ui->horizontalSlider,SIGNAL(sliderChanged(QObject\*)),this,SLOT(actByYourChange(QObject\*)));

connect(ui->pushButton,SIGNAL(clicked(bool)),ui->pushButton,SLOT(setbutton(bool)));

connect(ui->pushButton,SIGNAL(reset(QObject\*)),this,SLOT(actByYourChange(QObject\*)));

}

MainWindow::~MainWindow()

{   **delete** ui;

}

**void** MainWindow::actByYourChange(QObject\* sender){

if(senderObj==ui->comboBox){

QString size = ui->comboBox->currentText();

if(size=="0") {

ui->horizontalSlider->setSliderPosition(0);

ui->lineEdit->setText(size);

ui->progressBar->setValue(0);

}

if(size=="50") {

ui->horizontalSlider->setSliderPosition(50);

ui->lineEdit->setText(size);

ui->progressBar->setValue(50);

}

if(size=="100") {

ui->horizontalSlider->setSliderPosition(100);

ui->lineEdit->setText(size);

ui->progressBar->setValue(100);

}

}

else if(senderObj==ui->lineEdit){

QString size = ui->lineEdit->text();

ui->comboBox->setCurrentText(ui->lineEdit->text());

ui->horizontalSlider->setSliderPosition(size.toInt());

ui->progressBar->setValue(size.toInt());

}

else if(senderObj==ui->horizontalSlider){

int i = ui->horizontalSlider->value();

ui->comboBox->setCurrentText(QString::number(i));

ui->progressBar->setValue(i);

ui->lineEdit->setText(QString::number(i));

}

else if(senderObj == ui->pushButton){

ui->comboBox->setCurrentText(QString::number(0));

ui->progressBar->setValue(0);

ui->lineEdit->setText(QString::number(0));

ui->horizontalSlider->setslider(0);

}

ui->centralWidget->adjustSize();

}

**Mycombobox.h**

#ifndef MYCOMBOBOX\_H

#define MYCOMBOBOX\_H

#include <QComboBox>

**class** mycomboBox: **public** QComboBox {

Q\_OBJECT

**public**:

    mycomboBox(QWidget\* qw):QComboBox(qw){};

signals:

void **iChanged**(QObject\*); //My own signal

**public** slots:

void **myCurrentValueChanged**(const QString&);

};

#endif // /MYCOMBOBOX\_H

**Mycombobox.cpp**

#include "mycombobox.h"

void mycomboBox::**myCurrentValueChanged**(const QString&){

emit iChanged(this);

}

**MylineEdit.h**

#ifndef MYLINEEDIT\_H

#define MYLINEEDIT\_H

#include <QLineEdit>//box

**class** MyLineEdit:**public** QLineEdit{

Q\_OBJECT

**public**:

    MyLineEdit(**const** QString& qs):QLineEdit(qs){};

    MyLineEdit(QWidget\* qw):QLineEdit(qw){};

signals:

void **iChanged**(QObject\*); //My own signal

**public** slots:

void **myEditingFinished**();//For receiving its predefined signal. In its implementation, I emit my own signal.

};

 #endif // MYLINEEDIT\_H

**MylineEdit.cpp**

#include "mylineedit.h"

void MyLineEdit::**myEditingFinished**(){

emit iChanged(this);

}

**MySlider.h**

#ifndef MYSLIDER\_H

#define MYSLIDER\_H

#include <QSlider>//box here

**class** MySlider:**public** QSlider {

Q\_OBJECT

**public**:

    MySlider(QWidget\* qw):QSlider(qw){}

signals:

void **sliderChanged**(QObject\*); //My own signal

**public** slots:

void **setslider**(int size); // for the indirect signal-proxy-slot model

};

 #endif // MYSLIDER\_H

**MySlider.cpp**

#include "myslider.h"

void MySlider::**setslider**(int){

emit sliderChanged(this);

}

**Resetbutton.h**

#ifndef RESETBUTTON\_H

#define RESETBUTTON\_H

#include<QPushButton>

**class** ResetButton:**public** QPushButton  {

Q\_OBJECT

**public**:

    ResetButton(QWidget\* qw):QPushButton(qw){}

signals:

void **reset**(QObject\*); //My own signal

**private** slots:

void **setbutton**(bool); // for the indirect signal-proxy-slot model

};

#endif // RESETBUTTON\_H

**Resetbutton.cpp**

#include "resetbutton.h"

void ResetButton::**setbutton**(bool){

emit reset(this);

}

**(2) Sequence Diagram**

Consider the above application that uses a slider bar, progress bar, text input, combobox and a push button object. Draw a sequence diagram that depicts the following scenario: The main dialog window is open and a user changes slider position in main window.

**Solution:**

A screenshot of a social media post

Description automatically generated

Due: April 11, 11:59PM, 2019.

Turn in this word file with your answers via handin. The name of your word file should be: LastName\_FirstName.zip. For example, if your name is John Smith, you should turn in one file: Smith\_John.zip.