Segundo Certamen Práctico (Total Preg. 1 y 2: 120 + 15 minutos para subir archivos)
En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Pregunta 1:
Implemente la clase Matriz2x2 para representar matrices de 2x2 que incluya la sobrecarga de los siguientes operadores: *, +, <<. Su clase debe permitir su uso en el siguiente código:

```
#include "Matriz2x2.h"
#include <iostream.h>
using namespace std;
int main(void)
{
   Matriz2x2 A(2,4,5,7), B(-3,9,2,8), C;
   C=A*B;
   cout << C;
   cout << A+B;
}
```

Donde A y B están dadas por:

$$A = \begin{bmatrix} 2 & 4 \\ 5 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} -3 & 9 \\ 2 & 8 \end{bmatrix}$$

Recuerde que:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

Suba a aula los archivos Matriz2x2.h y Matriz2x2.cpp. Si gusta, puede utilizar el siguiente makefile:

```
# makefile
main: main.o Matriz2x2.o
    g++ -o main main.o Matriz2x2.o
main.o: main.cpp Matriz2x2.h
    g++ -c main.cpp
Matriz2x2.o: Matriz2x2.cpp Matriz2x2.h
    g++ -c Matriz2x2.cpp
clean:
    rm -rf *.o main
```

Solución: (Ver: códigos aquí)

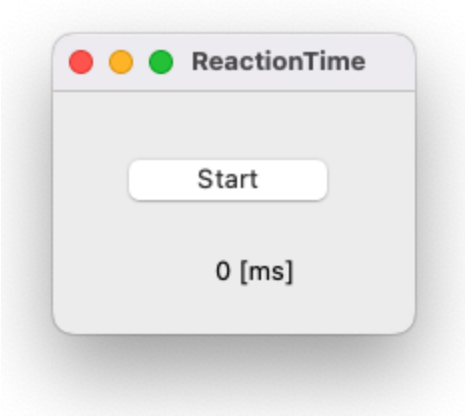
```
// Matriz2x2.h   Total: 34 pts.
#ifndef MATRIZ2x2_H
```

```
#define MATRIZ2x2_H
#include <iostream>
using namespace std;

class Matriz2x2 {
public: //  2 pts
   Matriz2x2(){  // 2 pts. También pudo dar valor inicial en sobrecargas
       for (int i = 0; i<2; i++)
           for (int j = 0; j<2; j++)
               values[i][j] = 0;
   }
   Matriz2x2(int a11, int a12, int a21, int a22){   // 5 pts. pudo ir en .cpp
       values[0][0] = a11;
       values[0][1] = a12;
       values[1][0] = a21;
       values[1][1] = a22;
   }
   Matriz2x2 operator * (const Matriz2x2 &) const;    // 6 pts.
   Matriz2x2 operator + (const Matriz2x2 &) const;    // 6 pts.
   friend ostream & operator<< (ostream &, const Matriz2x2 &); // 6 pts.
private:  // 2 pts.
       int values[2][2];  // pudo ser float u otro  // 5 pts.
};
#endif

// Matriz2x2.cpp   Total 16 pts.
#include "Matriz2x2.h"
Matriz2x2 Matriz2x2::operator * (const Matriz2x2 &B) const {   // 6 pts.
   Matriz2x2 temp;
   for (int k = 0; k<2; k++)
       for (int i = 0; i<2; i++)
           for (int j = 0; j<2; j++)
               temp.values[i][j] += values[i][k] * B.values[k][j];
   return temp;
}

Matriz2x2 Matriz2x2::operator + (const Matriz2x2 &B) const {  // 5 pts.
   Matriz2x2 temp;
   for (int i = 0; i<2; i++)
       for (int j = 0; j<2; j++)
           temp.values[i][j] = values[i][j] + B.values[i][j];
   return temp;
}

ostream & operator << (ostream &os, const Matriz2x2 &A) {  // 5 pts.
   for (int i = 0; i<2; i++) {
       for (int j = 0; j<2; j++)
           os <<" "<< A.values[i][j];
       os <<"\n";
   }
```

```
 return os;
}
```

Pregunta 2 (50 pts):

Desarrolle el programa Qt Reaction Time, el cual mide el tiempo en milisegundos entre dos veces en que se presiona un botón. Al partir el programa se muestra como la figura adjunta y su evolución sigue según se observa en este video. Notar que, al presionar por primera vez, el texto del botón cambia de Start a Stop. Luego de presionar "Start", éste cambia a "Stop" y el rótulo (label) cambia a "…. [ms]" mientras transcurre el tiempo. Al presionar "Stop", el rótulo muestra el tiempo en milisegundos desde que se presionó "Start". Esta secuencia se puede repetir hasta que la ventana sea cerrada.



Como respuesta suba a AULA un archivo comprimido (.rar, zip, o tar) del directorio donde Qt Creator deja los archivos fuentes de su proyecto (archivos .pro, .h, .cpp, y .ui).

Ayudas: Para medir el tiempo, revise la clase QTime, en particular considere sus funciones públicas restart() y elapsed(). Revise también la clase QString, en particular su miembro estático number(int n, int base=10).

Solución (ver códigos aquí)

```
// reactiontime.h    Total: 18 pts.
#ifndef REACTIONTIME_H
#define REACTIONTIME_H
#include <QWidget>
#include <QTime>
QT_BEGIN_NAMESPACE
namespace Ui { class ReactionTime; }
QT_END_NAMESPACE

class ReactionTime : public QWidget
{
   Q_OBJECT  // 5 pts. Macro + herencia de QWidget o QMainWindow

public:
   ReactionTime(QWidget *parent = nullptr);
   ~ReactionTime();
public slots:                    // 5 pts.
   void startStopTime();

private:
   Ui::ReactionTime *ui;
   QTime time;                  // 4 pts.
   bool timeRunning;            // 4 pts.
```

```
};
#endif // REACTIONTIME_H

// reactiontime.cpp   Total: 22 pts.
#include "reactiontime.h"
#include "ui_reactiontime.h"

ReactionTime::ReactionTime(QWidget *parent)
   : QWidget(parent)
   , ui(new Ui::ReactionTime)
{
   ui->setupUi(this);
   timeRunning=false;   // 4 pts.
   connect(ui->startButton,SIGNAL(clicked()),this,SLOT(startStopTime())); //
8 pts
}

void ReactionTime::startStopTime() {   // 10 pts.
   if(timeRunning) {
       ui->reactionLabel->setText(QString::number(time.elapsed())+" [ms]");
       ui->startButton->setText("Start");
   } else  {
       time.restart();
       ui->reactionLabel->setText("....  [ms]");
       ui->startButton->setText("Stop");
   }
   timeRunning=!timeRunning;
}

ReactionTime::~ReactionTime()
{
   delete ui;
}

// main.cpp  Total: 2 pts.
#include "reactiontime.h"
#include <QApplication>

int main(int argc, char *argv[])
{
   QApplication a(argc, argv);
   ReactionTime w;
   w.show();
   return a.exec();
}

// reactiontime.ui  8  pts. (4 QPushButton + 4 QLabel)
// se muestra una opción:
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ui version="4.0">
<class>ReactionTime</class>
<widget class="QWidget" name="ReactionTime">
 <property name="geometry">
  <rect>
   <x>0</x>
   <y>0</y>
   <width>181</width>
   <height>122</height>
  </rect>
 </property>
 <property name="windowTitle">
  <string>ReactionTime</string>
 </property>
 <widget class="QPushButton" name="startButton">
  <property name="geometry">
   <rect>
    <x>31</x>
    <y>30</y>
    <width>113</width>
    <height>32</height>
   </rect>
  </property>
  <property name="text">
   <string>Start</string>
  </property>
 </widget>
 <widget class="QLabel" name="reactionLabel">
  <property name="geometry">
   <rect>
    <x>20</x>
    <y>80</y>
    <width>101</width>
    <height>20</height>
   </rect>
  </property>
  <property name="text">
   <string>0 [ms]</string>
  </property>
  <property name="alignment">
   <set>Qt::AlignRight|Qt::AlignTrailing|Qt::AlignVCenter</set>
  </property>
 </widget>
</widget>
<resources/>
<connections/>
</ui>
```