Segundo Certamen

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.

Primera parte, sin apuntes (32 minutos; 32 puntos):

a) Se tiene la clase vector de la izquierda. Complete lo necesario para permitir los usos como el código de la derecha.

class CVector {
  int x, y;
  // ¿?
};
	CVector v1, v2;
	….
	if(v1 < v2)
	  cout << “la magnitud de v1 es menor que la de v2”;
	else
	  cout << “la magnitud de v1 no es menor que la de v2”;

Cómo mínimo se debe incluir la declaración e implementación de la sobrecarga del operador <.

```
class CVector {
   int x, y;
public:
   bool operator< (const CVector &v) const;    // 2 puntos
  // también será evaluado OK: bool operator< (CVector v);
};
```

En implementación se debe incluir: // +2 puntos

```
bool Cvector::operator<(const CVector &v) {
   return (x*x+y*y) < (v.x*v.x+v.y*v.y);
}
```

Obs: implementar como función inline es OK en este caso.

b) Entregue la declaración completa de la clase A y su implementación para lograr copia profunda y no generar fugas de memoria.

```
class A {
      double * f;
public:
        A() {
           f = new double[20];
        };
/* completar */
};
```

Se debería incluir constructor copia, sobrecarga operador asignación y destructor. Se dará puntaje total con atender dos de ellos correctamente.

```
class A {
   double * f;
public:
```

```
   A() {
        f = new double[20];
       };
   A(const &A a);
   A& operador= (const &A a) ;  // se considerará OK, A operator=(const A& a);
   ~A();
};  // 2 puntos. Uno por cada declaración
```

En implementación se debe incluir: // + 2 puntos. Uno por cada método

```
A::A(const &A a){
  f = new double[20];
  for(int i=0; i<20; i++)
    f[i] = a.f[i];
}
A& A::operador= (const &A a) {
  for (int i=0; i<20; i++)
    f[i] = a.f[i];
  return *this;
}
A::~A() {
 delete [] f;
}
```

c) Un desarrollador de software crea un Caso de Uso cuyo curso normal de eventos es:

Usuario	Aplicación
1) Ejecuta la aplicación	
	2) Muestra ventana de autenticación
3) Ingresa su usuario y clave.	
	4) Verifica que la clave corresponde al usuario y guarda la hora de acceso en un arreglo ubicado en memoria dinámica.
	5) Muestra la interfaz principal

¿Qué estaría de más en este caso de uso para una mejor descripción? Justifique.

Está de más señalar “en un arreglo ubicado en memoria dinámica.” // 2 puntos
Justificación: una decisión de implementación no debe ser incluida en un caso de uso. Esa parte de la descripción no es comprensible para un cliente. // +2 puntos

d) Explique cuándo es necesario utilizar declaraciones incompletas de clases y dé un ejemplo.
Las declaraciones incompletas se utilizan cuando existen dependencias circulares (2 puntos)

Ej: (2 puntos)

#include “B.h”	class A; // Declaración incompleta

class A{
B b;
};
	class B{
	A a;
	};

En este caso, existe una dependencia circular entre las clases A y B, la que puede ser resuelta con una declaración incompleta de A en el archivo "B.h"

e) Para el desarrollo de un proyecto, ud. y un compañero encuentran un código por internet que resuelve todos los problemas de su proyecto. Su compañero argumenta que el código encontrado, al no estar bajo ningún tipo de licencia, puede ser usado libremente.
Indique si lo dicho por su compañero es verdadero o falso. Justifique.

Falso (2 puntos), pues cualquier código sin licencia, se supone **con copyright**, por lo que nadie puede hacer uso de él sin permiso explícito del creador. (2 puntos)

f) Considere el siguiente código y responda

```
class C{
   g(){}
};

C var1; // Variable var1
C *var2 = new C; // Variable var2
```

- ¿cuál es la diferencia entre las variables *var1* y *var2*?

  var1 se encuentra en memoria estática/stack (dependiendo de donde sea creada), mientras que var2 es un puntero que apunta a un objeto almacenado de memoria dinámica. 2 puntos

- Indique la sintaxis para acceder al método *g()* a partir de las variable *var1* y *var2*

  var1.g();

  var2->g(); // (*var2).g(); 2 puntos

g) Para el código de la izquierda, indique si cada caso de la derecha arrojaría error. Si no genera error, indique el nombre de la clase cuyo método es invocado.

class A {
public: void m1() {};
virtual void m2() {};
};
class B: public A {
public: virtual void m1() {};
void m2() {};
};
class C: public B {
public: void m1() {};
	Caso 1)
	B b;
	A *a = &b;
	(*a).m1();
	
	Caso 2)
	C c;
	A &a = c;
	a.m2();

```
        void m2(){};
};
```

Ninguno de los dos casos arroja error.
Caso 1: Se ejecuta la implementación de m1 dada en A. // 2 puntos
Nota: Al no ser virtual no aplica ligado dinámico y se ejecuta la implementación de la clase del puntero (o referencia, no es este caso).
Caso 2: Se ejecuta la implementación de m2 dada en C. // +2 puntos
Nota: Al ser virtual aplica ligado dinámico y se ejecuta la implementación de la clase del objeto siendo referenciado (o apuntado, no es este caso)

h) Considere el siguiente código

```
class A{
  public:
    void f( ){ }
};

class B{
  public:
    void f( ){ }
};

class C : public A, public B
{  };

void main(){
    C objeto;
    objeto.f();
}
```

Indique si el siguiente código genera un error o no. Justifique.

Sí, genera error (2 puntos). Esto pues, debido a la herencia múltiple, el compilador de C++ no sabe a qué método f() se refiere cuando se llama desde cualquier instancia de la clase C. (2 puntos)

Segunda Parte, con apuntes (68 minutos) Responda en hojas separadas.
Para responder las preguntas de desarrollo usted puede reutilizar parte de los códigos disponibles en la página del curso u otras fuentes. En estos casos indicar la fuente usada. **Códigos innecesarios para la pregunta serán penalizados**.

2.- Pregunta 2, desarrollo con apuntes. (34 puntos)
a) (7 puntos) Para el código provisto, señale **cuáles serían los comandos para compilar cada archivo .cpp y el comando para generar el ejecutable.** Suba a AULA un archivo "2a.txt" con sus respuestas.

b) (15 puntos) Como resultado de la ejecución usando el archivo "plan.txt" provisto, se observa que el objeto juan de la función "main" fue alterado como consecuencia del cambio de nota de copiaJuan. **Modifique lo necesario para que el cambio de nota de copiaJuan no afecte a juan.**
En su respuesta en AULA, **suba sólo los archivos (.h y/o .cpp) que haya modificado.**

c) (12 puntos) Así como una Asignatura es un "Ramo", una Actividad como educación física o deporte también es un "Ramo". Cree e implemente la clase Actividad como clase hija de Ramo. Esta clase contiene el tributo "calificacion" de tipo char. Su valor puede ser 'A' (para aprobado) o 'R' (para reprobado).
En su respuesta en AULA, **suba sólo lo archivos asociados a esta nueva clase.**

a) Para compilar se puede usar: // 3 puntos
$ g++ -c PlanDeEstudios.cpp
$ g++ -c main.cpp

Para generar el ejecutable se puede usar: // 4 puntos
$ g++ -o ejecutable main.o PlanDeEstudios.o

Nota: estudiantes que no responden como se pide y no se distingue la compilación de cada archivo con la generación del ejecutable obtuvieron 4/7. Notar se pedía: comandos para compilar y luego comando para generar el ejecutable (**3 comandos**).
La compilación separada es un aprendizaje importante. La opción -c permite compilar un archivo único o varios. Otra cosa es generar un ejecutable a partir de varios archivos previamente compilados.
En proyectos grandes esto es importante. Cuando se modifica un programa es mala idea rehacer todas las compilaciones (puede tomar varios minutos) cuando bastaría recompilar sólo un archivo (pocos segundos) y luego ligar todo para generar el ejecutable. Por esto se pidió comandos para compilar cada archivo ….

Además de la importancia de este aprendizaje, también es importante responder las preguntas formuladas. Si se pide comandos para ... y luego el comando para generar el ejecutable, se está pidiendo tres comandos. Luego se dice "suba ... sus respuestas" (nuevamente plural).
Quizás sin comprender bien el proceso, algunos incluyeron el comando para ejecutar el programa. Eso no se pedía.

b) Se debe agregar un constructor (se aceptará si agrega sobrecarga de operador asignación)
En PlanDeEstudios.h // 5 puntos

```
#ifndef PLAN_DE_ESTUDIOS_H
#define PLAN_DE_ESTUDIOS_H
#include <vector>
#include <string>
#include "Ramo.h"
using namespace std;

class PlanDeEstudios {
private:
   vector<Ramo *> plan;
public:
   PlanDeEstudios(){};
   PlanDeEstudios(string file);
   PlanDeEstudios(const PlanDeEstudios& pe); // agregar aquí
   int getTotalCreditosAprobados();
   Ramo * operator[](int i);
   //PlanDeEstudios & operator= (const PlanDeEstudios& pe); // también será acpetado
};
#endif
```

En PlanDeEstudios.cpp // 10 puntos

```
#include <fstream>
#include <vector>
#include "PlanDeEstudios.h"
#include "Asignatura.h"

PlanDeEstudios::PlanDeEstudios(string fileName){
   int nRamos, creditos, nota;

   ifstream file(fileName);
   file >> nRamos;  // número de ramos del archivo
   for (int i=0; i< nRamos; i++){
      file >> creditos >> nota;
      plan.push_back(new Asignatura(creditos, nota));
   }
}
//////    PARTE QUE SE DEBE AGREGAR   ////////////
PlanDeEstudios::PlanDeEstudios(const PlanDeEstudios & pe){
   Asignatura * pa;
   for (int i=0; i < pe.plan.size(); i++) { // insertamos cada Ramo creando nuevos objetos
      pa = dynamic_cast<Asignatura *>(pe.plan[i]);
      if (pa != NULL) // hasta aquí siempre debería ser no NULL
         plan.push_back(new Asignatura(pa->getCreditos(),pa->getNota()));
```

```
    }
}
/// FIN DE PARTE AGREGADA  /////

int PlanDeEstudios::getTotalCreditosAprobados(){
    int nc=0;
    for (int i=0; i<plan.size(); i++)
        if (plan[i]->estaAprobada())
            nc+=plan[i]->getCreditos();
    return nc;
}

Ramo * PlanDeEstudios::operator[](int i){
    return plan[i];
}

/*
// También será aceptado
PlanDeEstudios & PlanDeEstudios::operator= (const PlanDeEstudios &pe){
    Asignatura * pa;
    cout << "sobrecarga...";
    while(plan.size() > 0) {
        delete plan.back();  // libera el último ramo
        plan.pop_back();     // remueve el último valor
    }
    for (int i=0; i < pe.plan.size(); i++) { // insertamos cada Ramo creando nuevos objetos
        pa = dynamic_cast<Asignatura *>(pe.plan[i]);
        if (pa != NULL) // hasta aquí siempre debería ser no NULL
            plan.push_back(new Asignatura(pa->getCreditos(),pa->getNota()));
    }
    return *this;
}
*/
```

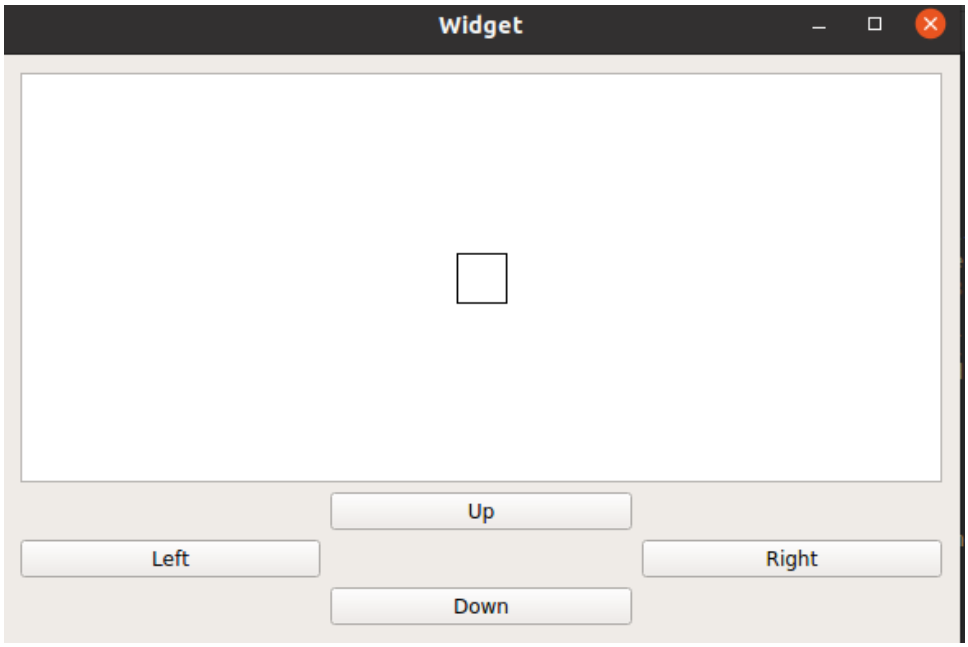
C) Se crea clase Actividad similar a Asignatura. En Actividad.h tenemos: // 12 puntos

```
 #ifndef ACTIVIDAD_H
#define ACTIVIDAD_H
#include "Ramo.h"

class Actividad: public Ramo {
private:
    char calificacion;
public:
```

```
    Actividad(int cred=0, char calif='R'): Ramo(cred), calificacion(calif){};
    char getCalificacion() { return calificacion;};
    void setCalificacion(char c) { calificacion=c;};
    virtual bool estaAprobada() { return calificacion=='A';};
};
#endif
```

3.- (34 puntos) Se desea construir un programa C++ usando las bibliotecas Qt que permita controlar los movimientos de una figura geométrica a través de distintos botones. Para ello se le ha entregado un código ya funcionando a través de la página de Aula. Este código le permite mover un rectángulo hacia arriba cada vez que se **mantiene presionado** el botón *up*. El movimiento se detiene cuando el botón *up* **se suelta.** La interfaz gráfica es la siguiente:



Debe agregar la siguiente funcionalidad al código entregado:
a) (17 pts.) Modifique el código entregado para que el rectángulo sea capaz de moverse además en las direcciones abajo, izquierda y derecha al mantener presionados los botones *down*, *left* y *right* agregados en el archivo *forms/widget.ui*. Cada vez que se suelta el botón respectivo, el movimiento debe parar.

Widget.h

```
class Widget : public QWidget
{
    Q_OBJECT

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

public slots:
    void released();
    void up();
```

```
    // Agregar slots adicionales (2 puntos)
    void down();
    void left();
    void right();

private:
    Ui::Widget *ui;
    QGraphicsScene *scene;
    RectangleMove *rect;
    QTimer *timer = new QTimer(this);
};
```

rectanglemove.h

```
class RectangleMove:public QObject,public QGraphicsRectItem{
    Q_OBJECT
public:
    RectangleMove(int x, int y, int w, int h);

public slots:
    void up();
    // Agregar slots adicionales (2 puntos)
    void down();
    void left();
    void right();
};
```

widget.cpp

```
Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget){
    ui->setupUi(this);
    scene = new QGraphicsScene(this);
    ui->graphicsView->setScene(scene);
    rect = new RectangleMove(0,0,30,30);
    scene->addItem(rect);
    connect(ui->up,SIGNAL(pressed()),this,SLOT(up()));
    connect(ui->up,SIGNAL(released()),this,SLOT(released()));

    // Agregar conexiones de signals y slots respectivas (5 puntos)
    connect(ui->down,SIGNAL(pressed()),this,SLOT(down()));
    connect(ui->left,SIGNAL(pressed()),this,SLOT(left()));
    connect(ui->right,SIGNAL(pressed()),this,SLOT(right()));
    connect(ui->down,SIGNAL(released()),this,SLOT(released()));
    connect(ui->left,SIGNAL(released()),this,SLOT(released()));
    connect(ui->right,SIGNAL(released()),this,SLOT(released()));
}
void Widget::up(){
    connect(this->timer,SIGNAL(timeout()),rect,SLOT(up()));
    this->timer->start(100);
}

// Implementar métodos (slots) down, left y right (4 puntos)
```

```
void Widget::down(){
    connect(this->timer,SIGNAL(timeout()),rect,SLOT(down()));
    this->timer->start(100);
}
void Widget::left(){
    connect(this->timer,SIGNAL(timeout()),rect,SLOT(left()));
    this->timer->start(100);
}
void Widget::right(){
    connect(this->timer,SIGNAL(timeout()),rect,SLOT(right()));
    this->timer->start(100);
}

void Widget::released(){
    disconnect(timer,SIGNAL(timeout()),nullptr,nullptr);
    this->timer->stop();
}
Widget::~Widget(){
    delete ui;
    delete scene;
    delete timer;
    delete rect;
}
```

rectangle.cpp

```
RectangleMove::RectangleMove(int posX, int posY, int ancho, int
alto):QGraphicsRectItem(posX,posY,ancho,alto)
{ }
void RectangleMove::up(){
    this->setY(this->y()-10);

}

// Implementar funciones down, left y right (4 puntos)
void RectangleMove::down(){
    this->setY(this->y()+10);
}
void RectangleMove::left(){
    this->setX(this->x()-10);
}
void RectangleMove::right(){
    this->setX(this->x()+10);
}
```

b) (17 pts.) Modifique el código anterior para que aparezca **un círculo** en vez de un rectángulo de tamaño 30x30 píxeles, el cual responda a los mismos movimientos ya definidos en la pregunta anterior. Utilice la clase *QGraphicsEllipseItem* definida dentro de las bibliotecas Qt para su desarrollo. **Entregue sus respuestas en un único archivo P3.zip (si lo desea, puede ser .tar) con todos los códigos de respuesta. En su entrega P3.zip, separe el desarrollo de cada respuesta en distintas carpetas.**

Cambios a realizar

circlemove.h

```
#include <QGraphicsEllipseItem>

// Crear clase circlemove (o equivalente) con los elementos necesarios (5 puntos)
class CircleMove:public QObject,public QGraphicsEllipseItem{
    Q_OBJECT
public:
    CircleMove(int x, int y, int w, int h);

public slots:
    void up();
    void down();
    void left();
    void right();

};
```

circlemove.cpp

```
// Implementar metodos de clase circlemove (o equivalente) (4 puntos)
CircleMove::CircleMove(int posX, int posY, int ancho, int
alto):QGraphicsEllipseItem(posX,posY,ancho,alto)
{}
void CircleMove::up(){
    this->setY(this->y()-10);
}
void CircleMove::down(){
    this->setY(this->y()+10);
}
void CircleMove::left(){
    this->setX(this->x()-10);
}
void CircleMove::right(){
    this->setX(this->x()+10);
}
```

widget.h

```
class Widget : public QWidget
{
    Q_OBJECT

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

public slots:
    void released();
    void up();
    void down();
    void left();
```

```
    void right();
private:
    Ui::Widget *ui;
    QGraphicsScene *scene;
    CircleMove *circle; // Crear variable tipo circlemove (1 punto)
    QTimer *timer = new QTimer(this);
};
```

widget.cpp

```
Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    scene = new QGraphicsScene(this);
    ui->graphicsView->setScene(scene);
    // Dejar variable circle apuntando a objeto circle move (o equivalente) (2 puntos)
    circle = new CircleMove(0,0,30,30);
    scene->addItem(circle);
    // Crear conexiones de signal y slot (2 puntos)
    connect(ui->up,SIGNAL(pressed()),this,SLOT(up()));
    connect(ui->down,SIGNAL(pressed()),this,SLOT(down()));
    connect(ui->left,SIGNAL(pressed()),this,SLOT(left()));
    connect(ui->right,SIGNAL(pressed()),this,SLOT(right()));
    connect(ui->up,SIGNAL(released()),this,SLOT(released()));
    connect(ui->down,SIGNAL(released()),this,SLOT(released()));
    connect(ui->left,SIGNAL(released()),this,SLOT(released()));
    connect(ui->right,SIGNAL(released()),this,SLOT(released()));
}
// Implementar métodos (slots) necesarios (2 puntos)
void Widget::up(){
    connect(this->timer,SIGNAL(timeout()),circle,SLOT(up()));
    this->timer->start(100);
}
void Widget::down(){
    connect(this->timer,SIGNAL(timeout()),circle,SLOT(down()));
    this->timer->start(100);
}
void Widget::left(){
    connect(this->timer,SIGNAL(timeout()),circle,SLOT(left()));
    this->timer->start(100);
}
void Widget::right(){
    connect(this->timer,SIGNAL(timeout()),circle,SLOT(right()));
    this->timer->start(100);
}
void Widget::released(){
    disconnect(timer,SIGNAL(timeout()),nullptr,nullptr);
    this->timer->stop();
}
Widget::~Widget()
{
    delete ui;
```

```
    delete scene;
    delete timer;
    delete circle; // Liberar circle de memoria (1 punto)
}
```