Primer 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** (40 minutos; 30 puntos):

a) Para cada caso del lado derecho indique la salida esperada o error de compilación producido. No considere errores por ausencia de archivos de encabezados o similar.

```
class A {
public:
  int x;
  A () {x=3;}
  int * get_px(); //se supuso *
  int & get_rx();
};

int * A::get_px(){
   return &x;
}

int & A::get_rx(){
   return x;
}
```

Caso a)
```
A v;
int *p = v.get_px();      (*p)++;
cout << "v.x=" << v.x << endl;
```

Caso b)
```
A v;
int &r = v.get_px();    r++;
cout <<  "v.x=" << v.x << endl;
```

Caso c)
```
A v;
v.get_rx() = 7;
cout << "v.x=" << v.x << endl;
```

Caso d)
```
A v;
A *p=&v;   v.x--;
cout << "p->get_rx()=" << p->get_rx()<<endl;
```

Lamentablemente se omitió el * en la declaración del método int * get_px( );
Si se supone que debía ir para consistencia con implementación, la respuesta es:
Caso a) v.x=4 Caso b) No compila, un puntero no puede ser referencia a un entero. Caso c) v.x=7 Caso d) p->get_rx()=2 // 2+1+1+1
Si se supone que debe sacarse el puntero en la implementación y sacar también &:
Caso a) No compila, Caso b) v.x=4 Caso c) v.x=7 Caso d) p->get_rx()=2.

b) Para el archivo b.h correspondiente a la declaración de la clase B, proponga una implementación simple para la clase B (b.cpp) que incluya un puntero no nulo para el atributo ***p***. (No agregue nuevos métodos ni constructores)

```
// b.h
class B {
private:
  const int id;
  static int id_global;
public:
    B();
   ~B();
    int * p;
    int getId();
}
```

```
#include "B.h"
int B::id_global=1;  // pudo ser cualquier valor inicial.   1pt.
```

```
B::B():id(id_global++) // 2 pts
{
  p = new int;
}
B::~B() {
   delete p;    //1 pt
}
int B::getId(){
    return id;        // 1pt
}
```

c) Considere el siguiente código:

```
class A{
        int x=0;
};

void showX(A a){
    cout << "El valor de x es: " << a.x << endl;
}
```

```
int main(){
    A p;
    showX(p);
    return 0;
}
```

Modifique el código de la ***clase A*** sin cambiar el nivel de acceso de su atributo ***x*** para que el código dentro de la función ***main*** se ejecute sin errores. Justifique. No considere errores por falta de cabeceras ni similares.

R:
Una función global no puede acceder a los atributos privados de una clase. Para que pueda acceder al atributo privado, debe definirse la función como *friend* de la clase A.

```
class A{
       int x=0;
    public:
       friend void showX(A a);
};
```

d) Explique qué es un memory leak (o fuga de memoria). Muestre con un ejemplo un código que produzca un memory leak y cómo resolverlo.

R: Un memory leak ocurre cuando un objeto en memoria dinámica es dejado sin ningún puntero para acceder a él.

Ej:

```
for(int i =0; i<=10; i++){
    // Como nunca se libera, en cada iteración esta memoria queda guardada y sin poder ser accedida
    Persona *p = new Persona();
}

// Se debe liberar la memoria cuando ya no se utiliza
```

```
for(int i =0; i<=10; i++){
   Persona *p = new Persona();
   delete p; // Fuera del for, la variable no está declarada
}
```

e) Suponga que, para una aplicación para fomentar el reciclaje, usted necesita usar una librería hecha por un tercero para procesar imágenes. Mencione y justifique alguna licencia que debería tener la librería para que usted pueda usarla y modificarla con el fin de agregar más filtros. Su finalidad es el uso solo en su aplicación y no tiene intenciones de crear una librería derivada.

Debería apuntar a describir cualquier licencia del tipo permisiva, particularmente MIT o GNU. Mitad de puntaje si solo habla de WTFPL, dado que ese sería un caso ideal. La idea es descubrir el tipo de licencia que al menos se debería considerar.

f) Se tiene la siguiente definición:

```
class Persona {
public: void displayNombre();
}
class Estudiante: protected Persona {
public: void displayNombre();
}
```

Para la siguiente implementación:

```
Estudiante *student = new Estudiante();
Persona *person;
person = student;
person->displayNombre(); // <-- [1]
```

Mencione y justifiqué qué modificador se debería agregar para que el código en [1] ejecute la implementación de displayNombre() de la clase Estudiante.

Se debe agregar el modificador virtual en la declaración del método displayNombre en la clase Persona.

Preguntas de Desarrollo:

2) Queremos almacenar estudiantes de ingeniería (E_ingenieria) y de posgrado (E_posgrado) en un único vector de punteros a Estudiantes. Los estudiantes de ingeniería tienen una carrera y para los de posgrados se registra su número de publicaciones.

Se cuenta con la declaración de la clase Estudiante la cual usted **no debe modificar**. Se le pide implementar la clase Estudiante (estudiante.cpp) (8 pts.) y declarar (.h) e implementar (.cpp) las clases E_ingenieria (12 pts.) y E_posgrado (15 pts.) de manera que el código **main dado** arroje como salida:

Juan es estudiante de Ing. Civil Tel.
Claudia es estudiante de posgrado con 2 publicaciones.
Claudia ha publicado más.

Usted debe subir un archivo con todo su proyecto Qt. Como nombre use: P1_ROL_NOMBRE_APELLIDO.zip, (Ej.: P1_12345-3_AGUSTIN_GONZALEZ.zip)

Puede acceder a los códigos en: http://profesores.elo.utfsm.cl/~agv/elo329/1s21/C2

```
#include <QCoreApplication>
#include <iostream>
#include "estudiante.h"
#include "e_ingenieria.h"
#include "e_posgrado.h"
using namespace std;
int main(int argc, char *argv[])
{
  QCoreApplication a(argc, argv);
  vector <Estudiante *> comunidad;
 E_ingenieria juan( "Juan", "Ing. Civil Tel");  // nombre, carrera
 E_posgrado claudia ("Claudia", 2);  // nombre, número de publicaciones
 comunidad.push_back(&juan);
 comunidad.push_back(&claudia);
 for (unsigned int i=0; i < comunidad.size(); i++)
        cout << *comunidad[i] << endl;

// El código siguiente sólo impacta en clase E_Posgrado
 E_posgrado pedro("Pedro", 1);
  if (pedro < claudia)
       cout << claudia.getNombre() << " ha publicado más." << endl;
 else
       cout << pedro.getNombre() << " ha publicado al menos lo mismo que " <<
            claudia.getNombre() << endl;
 return a.exec();
}
```

Solución:
Los archivos usted los puede encontrar en
http://profesores.elo.utfsm.cl/~agv/elo329/1s21/C2/solucion/

```
/*8 pts. estudiante.cpp*/
 #include "estudiante.h"
ostream & operator<<(ostream & os, const Estudiante & e){
       os << e.getDescription();    return os;
}
```

```
/* 6 pts. e_ingenieria.h*/
#ifndef E_INGENIERIA_H
#define E_INGENIERIA_H
#include "estudiante.h"
using namespace std;
class E_ingenieria: public Estudiante  // 1 {
public:
    E_ingenieria(string name, string carrera);  // 2
    virtual string getDescription() const;  //2
private:
   string carrera;  //1
};
#endif // E_INGENIERIA_H

/* 6 pts. e_ingenieria.cpp*/
#include "e_ingenieria.h"
E_ingenieria::E_ingenieria(string name, string carrera_):Estudiante(name)  // 3
{
  carrera= carrera_;
 }
string E_ingenieria::getDescription() const {  // 3
   return getNombre()+ " es estudiante de "+carrera+".";
}

/* 7 pts. e_posgrado.h*/
#ifndef E_POSGRADO_H
#define E_POSGRADO_H
#include "estudiante.h"
class E_posgrado: public Estudiante  // 1
{
public:
   E_posgrado( string name, int numPub);   // 2
  virtual string getDescription() const;  // 1
   bool operator<(const E_posgrado &ep);  // 2
private:
    int numPublicaciones;    //1
};
#endif // E_POSGRADO_H

/* 8 pts. e_posgrado.cpp*/
#include "e_posgrado.h"
E_posgrado::E_posgrado(string name, int np):Estudiante(name)  // 2
{
    numPublicaciones=np;
}
string E_posgrado::getDescription() const {     // 3
   return getNombre() + " es estudiante de posgrado con "+          to_string(numPublicaciones)+"
publicaciones.";
```

```
}
bool E_posgrado::operator<(const E_posgrado &ep){   // 3
   return numPublicaciones < ep.numPublicaciones;
}
```

3)

Instrucciones

- Entregue en la página de aula un solo archivo comprimido que contenga todos los archivos de su proyecto Qt.
- El nombre del archivo debe tener el siguiente formato: ROL_NOMBRE_APELLIDO.zip, sin tildes. (Ej. 2530009-2_PATRICIO_OLIVARES.zip)

Considere el código C++ sobre bibliotecas Qt disponible en el repositorio gitlab https://gitlab.com/patricio.olivaresr/elo329-2021-c2-p3.git.

Dicho código genera un scatter plot, donde la posición de cada punto *(x, y)* dentro del gráfico se determina de manera aleatoria. Los valores para las coordenadas x e y quedan determinadas por el rango de valores dado por $x, y \in [0, 1]$. Los puntos van apareciendo en el gráfico cada 1000 milisegundos (1 segundo).

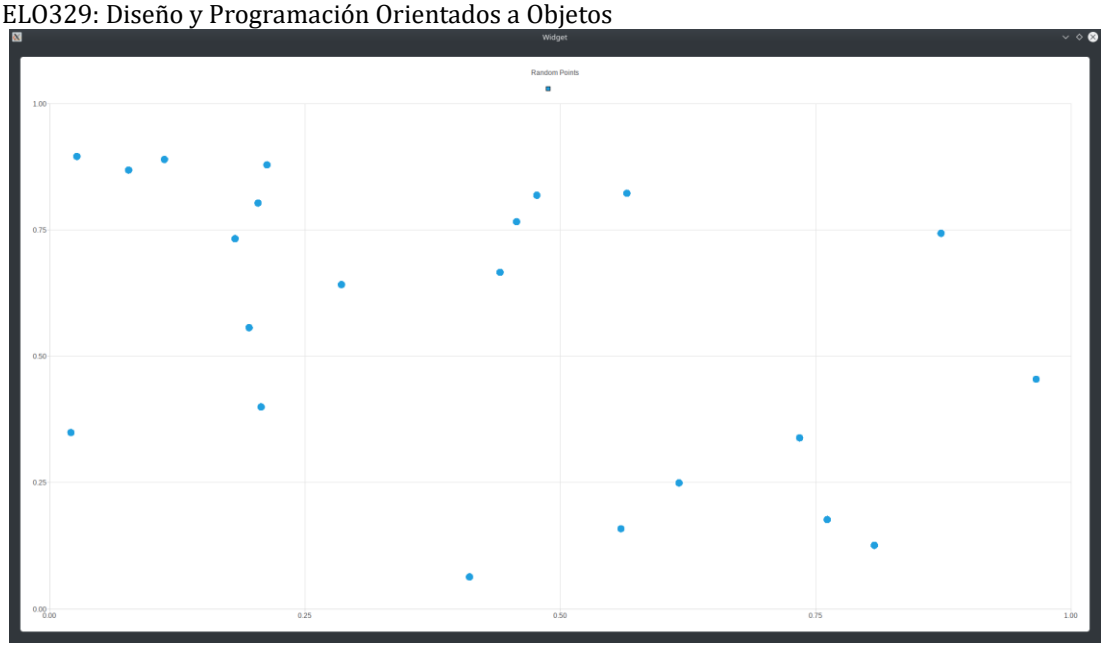

Figura 1: Captura del código disponible en gitlab. Los puntos van apareciendo cada 1 segundo en el gráfico.

1. (20pts) Agregue un botón de *start* y botón de *stop*, cuya función será iniciar y detener la agregación de nuevos puntos al gráfico respectivamente

R: Luego de agregar los botones a la interfaz gráfica, es necesario agregar dos slots en el código: uno para iniciar el timer y otro para detenerlo:

- Agregar los botones gráficos (5pts)
- Agregar slots para inicio y detención y sus respectivas implementaciones (10pts)

```
/*Widget.h*/
// 5pts
public slots:
        void start();
        void stop();

/*Widget.cpp*/
// 5pts
```

```
void Widget::start(){
/* this->time corresponde al tiempo definido
  * para la solución de la parte 2 de esta pregunta.
* Si en este punto, solo tiene definido 1000ms fijos
 * como tiempo, igualmente puede tener su puntaje
  * completo     */
   timer->start(this->time);
}
void Widget::stop(){
  timer->stop();
}
```

- Conectar signals de cada botón con su respectivo slot (5pts)

```
/*Widget.cpp*/
 connect(ui->start, SIGNAL(pressed()), this, SLOT(start()));
connect(ui->stop, SIGNAL(pressed()), this, SLOT(stop()));
```

2. (15pts) Agregue un *slider* que permita modificar el tiempo en el que los puntos van apareciendo en el gráfico. Este *slider* debe cumplir lo siguiente:

1. (5pts) El *slider* se debe mover entre 100 milisegundos (0.1 segundos) y 2000 milisegundos (2 segundos).

R:

- Agregación de slider gráfico (1pts)
- Definir mínimo y máximo del slider entre 100 y 2000 (1pts)
- Almacenar el tiempo (1pts)

```
/*Widget.h*/
private:
  int time=1000;
```

- Crear slot para modificar el tiempo (1pts)

```
/*Widget.h*/
public slots:
  void changeTime(int);

/*Widget.cpp*/
void Widget::changeTime(int time){
  this->time = time;
}
```

- Conectar signal de slider con slot de modificación de tiempo (1pts)

```
/*Widget.cpp*/
connect(ui->horizontalSlider,SIGNAL(valueChanged(int)), this, SLOT(changeTime(int)));
```

2. (5pts) Debe ir acompañado con un *display* LCD que muestre la cantidad de milisegundos Seleccionados.

R:

- Agregar display LCD (2pts)
- Conectar display LCD con slider (puede ser gráficamente o por código) (3pts)

3. (5pts) El tiempo solo debe ser actualizado una vez se presione nuevamente el botón *start*.

R:

- Agregar variable que almacena el tiempo al iniciar el timer (5pts)

```
/*Widget.cpp*/
void Widget::start(){
  timer->start(this->time);
}
```

Puede verificar el funcionamiento final del programa en el video disponible en el siguiente enlace: https://youtu.be/9h9TC3DV5yA