Nombre: _____

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) Para cada caso, en la columna derecha indique qué se imprime con el segmento de código central.

<table>
<tr><td rowspan="4">class Cvector {
public:
  int x, y;
 Cvector () {x=4; y=3;};
 int & getX();
};

int & Cvector::getX(){
  return x;
}</td><td>Cvector v;
Cvector * pv = &v; v.x++;
cout << “pv -> x=” << pv-> x << endl;</td><td>pv-> x= 5</td></tr>
<tr><td>Cvector v;
Cvector &rv=v; v.x++;
cout << “rv.x=” << rv.x << endl;</td><td>rv.x= 5</td></tr>
<tr><td>Cvector v;
 int a = v.getX(); v.x++;
cout << “a=” << a <<”, v.x=”<<v.x<< endl;</td><td>a=4, v.x=5</td></tr>
<tr><td>Cvector v;
 int &a = v.getX(); v.x++;
cout << “a=” << a <<”, v.x=”<<v.x<< endl;</td><td>a=5, v.x=5</td></tr>
</table>

b) Inserte en las secciones solicitadas (notar en ambos lados) el código necesario para liberar la memoria utilizada en el *heap*, evitando fugas por dejar memoria fuera de alcance.

```
class Image
{
public:
    Image(int w, int h);
    ~Image();
private:
    char *pixel_r;
    char *pixel_g;
    char *pixel_b;
};


Image::~Image()
{      // COMPLETAR

   delete [] pixel_r;
   delete [] pixel_g;
   delete [] pixel_b;

}
```

```
Image::Image(int w, int h) {
    char *tmpR = new char;
    char *tmpG = new char;
    char *tmpB = new char;

    pixel_r = new char[w*h]();
    pixel_g = new char[w*h]();
    pixel_b = new char[w*h]();


    for (int wi = i; wi < w; wi++) {
        …


    }

// COMPLETAR

   delete tmpR;
   delete tmpG;
   delete tmpB;

}
```

Nombre: _____

c) Considere las clase *cuadrado* y *rectangulo*. En *cuadrado* se incluye *rectangulo* como clase amiga (*friend*). Indique si las secciones de código de los 2 cuadros de la derecha son correctos, e indicar motivo del error en caso contrario.

```
class rectangulo;
class cuadrado
{
public:
    cuadrado(int a);
    bool
compararArea(rectangulo r);
    ~cuadrado();
friend class rectangulo;
private:
    int a;
};

class rectangulo
{
public:
    rectangulo(int a, int b);
    bool compararArea(cuadrado
r);
    ~rectangulo();
private:
    int a;
    int b;
};
```

```
bool cuadrado::compararArea(
         rectangulo r) {
    return a*a>r.a*r.b;
}
```

Incorrecto.

La clase cuadrado no puede acceder a los atributos privados de rectangulo. Si rectangulo declarase amistad con cuadrado, cuadrado podría manipular sus atributos privados.

```
bool rectangulo::compararArea(
           cuadrado c) {
    return a*b>c.a*c.a;
}
```

Correcto

Al ser cuadrado friend de la clase rectangulo, rectangulo tiene acceso a los atributos privados de cuadrado.

d) Considerando las definiciones de clases de la izquierda, indique qué se imprime al correr el cada segmento de código código central

```
class Persona {
public:
     Persona() {edad=20;};
     virtual int getEdad();
protected:
     int edad;
};
int Persona::getEdad(){
  return edad;
}
class Estudiante : public Persona {
 public:
   int getEdad();
};
int Estudiante::getEdad(){
   return 18;
}
```

Estudiante e; Persona p = e; cout << “Edad =” << p.getEdad();	**Edad=20**
Estudiante e; Persona *pp = &e; cout<<“Edad =”<< pp->getEdad();	**Edad=18**
Estudiante e; Persona &rp=e; cout << “Edad =” << rp.getEdad();	**Edad=18**
Estudiante *pe = new Estudiante(); Persona &rp = *pe; cout << “Edad =” << rp.getEdad();	**Edad=18**

Nombre: _____

e) Complete el código que define el método processingUnit **getWorstUnit**(). Este método debe retornar la unidad del vector *units* que presente menor eficiencia.
Optional Hint: <algorithm> contiene std::min_element(iterator start, iterator end);

```
#include <vector>
#include <algorithm>

using namespace std;
class processingUnit {
 public:
    processingUnit(float ef);
    bool operator<(const processingUnit & pU) {
        return eficacy < pU.eficacy;
    }
 private:
    float eficacy;
};

class processingSet {
 public:
    processingSet();
    ~processingSet();
    processingUnit getWorstUnit();
 private:
        vector<processingUnit> units;
};
```

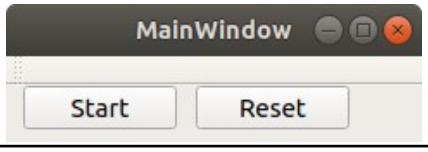
```
processingUnit processingSet::getWorstUnit() {

// COMPLETAR

 if (units.empty())
       return processingUnit(0.0);
   // Usando for
   processingUnit u = units[0];
   for (int i=1; i< units.size(); i++) {
       if (units[i] < u)
           u = units[i];
   }
   return u;
   // Usando min_element
   return *min_element(units.begin(),
               units.end());


}
```

f) Introduzca en el recuadro de la derecha el código necesario para que cuando se presione el botón *start* comience a correr el timer *t1*. Además, cuando termine el timer o se presione el botón *reset,* se debe detener el timer y ejecutar el slot *doReset()*. Los botones *start* y *reset* son parte de la interfaz de la ventana principal (*MainWindow*).



Hints: La clase QTimer cuenta con la señal *timeout()* y slots *start()* y *stop()*.
La clase QPushButton cuenta (entre otros) con las señales pressed*(), released()* y *toggled(),*y el slot *click()*.

```
...

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

    QTimer *t1 = new QTimer(this);
    t1→setInterval(3000);

    // COMPLETAR

}

void MainWindow::doReset(){
    std::cout << "Reset!!" << std::endl;
}
```

```
 // Inicia timer al presionar start
connect(ui->start, SIGNAL(pressed()),
        t1,SLOT(start()));

// Presiona botón reset al terminar el timer
connect(t1, SIGNAL(timeout()),
        ui->reset,SLOT(click()));

// Presiona botón reset al terminar el timer
connect(ui->reset,SIGNAL(pressed()),
        t1,SLOT(stop()));

// Ejecuta reset cuando se presiona el botón
// o termina el timer (connect anterior)
connect(ui->reset,SIGNAL(pressed()),
        this,SLOT(doReset()));
```

Nombre: _____

h) Para la clase Complejo, qué prototipo e implementación incorporaría usted para permitir instrucciones del tipo: Complejo z1(2,5); Complejo z2 =3*z1;

```
class Complejo {
private:
   float r, i;
public:
   Complejo();
   Complejo(float , float );
   float getRe();
   float getIm();
};
Complejo::Complejo() {r=i=0.0;}
Complejo::Complejo(float x, float y) {
   r=x;   i=y;
}
float Complejo::getRe() { return r;}
float Complejo::getIm() {return i;}
```

Ponga aquí el o los prototipos que debe agregar a la clase.

```
friend Complejo operator*(int, const Complejo&);
```

Ponga aquí la o las implementaciones que debe agregar:

```
Complejo operator* (int f, const Complejo & c){
   return Complejo(f*c.r, f*c.i);
}
```

j) En los niveles de certificación CMM (Capability Maturity Model), mencione característica importante presente en el nivel 4 y no en el nivel 3.

En nivel 4 se espera que la organización efectúe mediciones del proceso para así lograr un proceso controlado.

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.- (34 puntos) Usted trabajará aquí con variantes a la primera etapa de la Tarea 3. Recuerde que en esta etapa, no hay temporizadores ni salidas en tiempo real.

Puede ver el código en: http://profesores.elo.utfsm.cl/~agv/elo329/1s19/C2/P2

a) Se tiene el código de una solución para esta etapa, en la cual se desea que la variable tiempo que maneja la función main sea accesible para toda instancia de TrafficLight. ¿Qué modificación debe hacer usted y a qué archivo para cumplir ese requerimiento? La versión actual del código arroja (sin avanzar el tiempo):

```
agustin@agustin:~/WWW/elo329/1s19/C2/P2$ ./P2 2  3
0       V
0       A
0       R
0       V
0       A
0       R
```

(10 puntos) Se debe modificar archivo TrafficLight.h:

```
class TrafficLight {
public:
   TrafficLight (ostream &, int &, int ft, int tt);
```

Nombre: _____

```
   void turnStop();
   void turnTransition();
   void turnFollow();
   int getFollowTime();
   int getTransitionTime();
   TrafficLightState getState();
   virtual void printState()=0;
protected:
   ostream &os;
   int & time; //<<-- único cambio requerido: int time; → int & time;
private:
   int followTime;
   int transitionTime;
   TrafficLightState state;
};
```

b) A partir del código entregado, señale qué modificación debe hacer usted para crear la clase para semáforos peatonales: CrosswalkTrafficLight. En esta clase se pide que la salida sea "R" para luz roja, "R/V" para denotar parpadeo, y "V" para verde. Se pide su archivo .h y archivo .cpp para esta clase.

(12 puntos)

CrosswalkTrafficLight.h (6 puntos):

```
#ifndef CROSSWALK_TRAFFIC_LIGHT_H
#define CROSSWALK_TRAFFIC_LIGHT_H
#include "TrafficLight.h"
using namespace std;
class CrosswalkTrafficLight: public TrafficLight {
public:
    CrosswalkTrafficLight (ostream &, int &time, int ft, int tt);
    virtual void printState ();
};
#endif
```

CrosswalkTrafficLight.cpp (6 puntos)

```
#include "CrosswalkTrafficLight.h"
CrosswalkTrafficLight::CrosswalkTrafficLight (ostream & _os, int &t,
                                              int ft, int tt):
   TrafficLight(_os, t, ft, tt) {
}
void CrosswalkTrafficLight::printState () {
   switch (getState()) {
      case TrafficLightState::STOP: os << time << "\tR" << endl;
                break;
      case TrafficLightState::TRANSITION: os << time << "\tR/V" << endl;
                break;
      case TrafficLightState::FOLLOW: os << time << "\tV" << endl;
                break;
   }
}
```

Nombre: _____

c) Supongamos que deseamos tener un número arbitrario, nSc, de semáforos de calle y un semáforo peatonal, de manera que todos los semáforos de calle toman su turno en secuencia uno después del otro y finalmente toma su turno el semáforo peatonal. Modifique la función main de manera que reciba dos parámetros: número de ciclos (N) y número de semáforos de calle (nSc); y muestre la salida de los N ciclos para los nSc semáforos de calle (todos ellos con iguales tiempos de verde y amarillo). En este caso la invocación del programa es: $ ./P2 <N> <nSc>

(12 puntos)

```
#include "StreetTrafficLight.h"
#include "CrosswalkTrafficLight.h"
#include <iostream>   //cout
#include <vector>
using namespace std;
int main (int argc, char * argv[]) {
   int nCiclos=atoi(argv[1]);
   int nScalle=atoi(argv[2]);
   int time=0;  // tiempo en segundos
   vector<TrafficLight*> semaforos;
   for (int i=0; i<nScalle; i++)
       semaforos.push_back(new StreetTrafficLight(cout, time, 4, 2));
   semaforos.push_back(new CrosswalkTrafficLight(cout, time, 4, 2));
   for (int n=0; n < nCiclos; n++) {
       for (int i=0; i<semaforos.size(); i++) {
           semaforos[i]->turnFollow();
           time+=semaforos[i]->getFollowTime();
           semaforos[i]->turnTransition();
           time+=semaforos[i]->getTransitionTime();
           semaforos[i]->turnStop();
        }
   }
}
```

3.- (34 puntos) Deberá completar el código de un repositorio de recetas de cocina.
http://profesores.elo.utfsm.cl/~agv/elo329/1s19/C2/P3
Cada receta considera nombre, ingredientes, y un texto con las tareas y procedimientos que describe la receta. Las recetas se pueden crear y guardar, o leer.

a) Incluya las condiciones try-catch necesarias para evitar cualquier error de manipulación de archivos (utilizados para guardar/leer recetas).

```
Fuera del alcance de esta pregunta, para habilitar las excepciones en el uso de
ficheros se debe configurar el fstream a través del método exceptions:

  ofstream archivo;
  archivo.exceptions ( std::ifstream::failbit | std::ifstream::badbit );

Las condiciones try-catch necesarias son: encerrar en bloque try código contenido en

   receta::receta(string nombre)
   receta::~receta()
   receta::crear()
```

Nombre: _____

Incluir catch para problemas de IO (u otro catch que capture estas excepciones):

```
  catch (std::ios_base::failure& e) {
    std::cerr << e.what() << '\n';
  }
```

El código resultante es:

```
#ifndef RECETA_H
#define RECETA_H

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;

class receta {
public:
  receta(string nombre);
  ~receta();
  void leer();
  static void crear();

private:
  fstream archivo;
  string nombre;
  vector<string> ingredientes;
  string instrucciones;
};

receta::receta(string nombre) {
  string aux;

  // Habilita el uso de excepciones sobre el manejo de ficheros
  archivo.exceptions ( std::ifstream::failbit | std::ifstream::badbit );

  // Funciones de lectura son encerrados en try
  try {
    archivo.open (nombre+".receta");
    this->nombre = nombre;
    getline(archivo, nombre);
    getline(archivo, aux); //
    if (aux == "ingredientes:") {
      getline(archivo, aux); //
        while(aux != "instrucciones:") {
          ingredientes.push_back(aux);
          getline(archivo, aux); //
        }
      getline(archivo, instrucciones); //
      }
  }
  catch (std::ios_base::failure& e) {
    std::cerr << e.what() << '\n';
  }
}
```

Nombre: _____

```
receta::~receta() {
  // Try para intentar cerrar el archivo
  try {
    archivo.close();
  catch (std::ios_base::failure& e) {
    std::cerr << e.what() << '\n';
  }
}

void receta::leer() {
  cout << "Receta " + nombre << endl << endl;
  cout << "Ingredientes: " << endl;
  for(int i=0; i<ingredientes.size(); i++) {
    cout << ingredientes[i] << endl;
  }
  cout << endl;
  cout << "Instrucciones:" << endl << instrucciones << endl;
}

void receta::crear() {
  ofstream archivo;
  string nombre;

  archivo.exceptions ( std::ifstream::failbit | std::ifstream::badbit );

  cout << "Indique el nombre de la receta:" << endl;
  getline(cin, nombre);
  try { // Nueva manipulación de ficheros
    archivo.open(nombre+".receta");
    archivo << nombre << endl;

    cout << endl << "Ingrese ingredientes: (vacio para terminar)" << endl;
    string ingrediente;
    archivo << "ingredientes:" << endl;
    while(true) {
      getline(cin, ingrediente);
      if (!ingrediente.empty())
        archivo << ingrediente << endl;
      else
        break;
    }
    cout << "Ingrese instrucciones: (una única entrada de texto)" << endl;
    string instrucciones;
    getline(cin, instrucciones);
    archivo << "instrucciones:" << endl << instrucciones << endl;
    archivo.close();
  }
  catch (std::ios_base::failure& e) {
    std::cerr << e.what() << '\n';
  }
}
#endif // RECETA_H
```

b) Modifique la clase ingrediente tal que en vez de guardar un nombre, guarde la referencia a un archivo del ingrediente. Si el ingrediente no existe previamente el archivo debe crearse. Considere el archivo

Nombre: _____

“tomate.ingr” como ejemplo.

Durante la prueba se informó de la siguiente aclaración:
“Hay un error en pregunta 3b (se pide crear clase ingredientes).
Reeemplace el vector<string> de ingredientes de la clase receta en receta.h por un vector<ingrediente>. Debe crear la clase ingrediente, añadiéndole los atributos suficientes para guardar el contenido presentado en el archivo de ejemplo tomate.ingr."

Modificación en la clase receta (receta.h)

```
class receta {
public:
   receta(string nombre);
   ~receta();
   void leer();
   static void crear();

private:
   fstream archivo;
   string nombre;
   vector<ingrediente*> ingredientes;
   string instrucciones;
};

receta::receta(string nombre) {
  string aux;
  archivo.open (nombre+".receta");
  this->nombre = nombre;
  getline(archivo, nombre);
  getline(archivo, aux); //
  if (aux == "ingredientes:") {
   getline(archivo, aux); //
   while(aux != "instrucciones:") {
                        // Carga el ingrediente en el vector de ingredientes
      ingredientes.push_back(new ingrediente(aux, nombre));
      getline(archivo, aux); //
   }
   getline(archivo, instrucciones); //
  }
}
```

Clase ingrediente.h

```
#ifndef INGREDIENTE_H
#define INGREDIENTE_H

#include <iostream>
#include <fstream>
#include <string>
#include <vector>

using namespace std;
```

Nombre: _____

```
class ingrediente {
   // Código solicitado: clase ingrediente con constructor o método de
lectura de ingredientes
   // levanta información siguiendo archivo de ejemplo tomates.ingr
   // ***************
private:
  fstream archivo; // artibuto que permite guardar referencia a archivo
  string nombre;   // artibuto que permite guardar nombre de ingrediente
  vector<string> tips;  // artibuto que permite guardar tips según archivo
ejemplo
  vector<string> recetas; // artibuto que permite guardar recetas según
archivo ejemplo

public:
   ingrediente(string nombre, string receta) {
     string aux;
     bool added = false;
      this->nombre = nombre;

       archivo.open (nombre+".ingr");
       if(archivo) {
          getline(archivo, aux); //
          if (aux == "tips") {
            getline(archivo, aux); //
              while(aux != "recetas") {
                tips.push_back(aux);
                getline(archivo, aux); //
              }
              while(!archivo.eof()) {
                getline(archivo, aux); //
                if(!aux.empty())
                  recetas.push_back(aux);
                if (aux == receta) {
                  added = true;
                  continue;
                }
              }
            }
        }
        archivo.close();
      if(!added) {
        recetas.push_back(receta);
      }
      save();
   }

// *************

// A continuación se añaden métodos para completar su uso dentro de
recetario.cpp
// Estos métodos no forman parte de la evaluación.
```

Nombre: _____

```
    ~ingrediente() {
       if(archivo.is_open())
          archivo.close();
       save();
    }

    string getName() {
       return nombre;
    }

    void getTips() {
       cout << "\tTips para " + nombre + ":" << endl;
       if (!tips.size())
          cout << "\t\tNinguno" << endl;
       for(int i=0; i<tips.size(); i++) {
          cout << "\t\t" << tips[i] << endl;
       }
    }

    void getRecetas() {
       cout << "\tRecetas donde se usa(n) " + nombre + ":" << endl;
       if (!recetas.size())
          cout << "\t\tNinguna" << endl;
       for(int i=0; i<recetas.size(); i++) {
          cout << "\t\t" << recetas[i] << endl;
       }
    }

    void save() {
       ofstream archivo;
                archivo.open(nombre+".ingr");

              archivo << "tips" << endl;
           for(int i=0; i<tips.size(); i++) {
             archivo << tips[i] << endl;
       }
                archivo << "recetas" << endl;
       for(int i=0; i<recetas.size(); i++) {
          archivo << recetas[i] << endl;
       }
                archivo.close();
        }
};
#endif // INGREDIENTE_H
```