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):

1.- Responda brevemente y entregue en hoja con su nombre. Use buena letra.

a) ¿Cuál es el propósito de los atributos "codebase" y "archive" del rótulo <applet>?
Codebase: permite especificar la ruta respecto de la ubicación del HTML donde se encuentra los archivos del applet.
Archive: Permite listar el conjunto de .jar y otros recursos requeridos por la aplicación.

b) Si en una entrevista de trabajo a usted le preguntan qué es la Ingeniería de Software ¿qué diría?
La ingeniería de software es una disciplina dedicada al diseño, construcción y mantención de programas computacionales eficientes, económicos que logran cumplir sus objetivos en tiempos definidos.

c) Además de "identificar una oportunidad de negocio" y "levantamiento inicial de requerimientos", mencione 4 otras etapas de un proyecto de software.
Confección de propuesta, Análisis del problema, Definición de la arquitectura, Diseño de la solución, Codificación de la solución, Pruebas Unitarias, Pruebas de Integración, Capacitación de los usuarios, Marcha blanca, Puesta en producción.

d) Mencione un tipo de diagrama UML de estructura y un tipo de diagrama UML de comportamiento.
Diagrama de estructura: Diagrama de clases.
Diagrama de comportamiento: Diagrama de secuencia, diagrama de estados.

e) Indique qué elemento diferenciador agrega cada uno de los 3 niveles superiores de certificación CMM (Modelo de Madurez de la Capacidad) respecto del previo. Ej. Nivel 2 es capaz de *repetir* éxitos previos (esto no lo hace el nivel 1).
Nivel 3 tiene un proceso definido que no tiene el Nivel 2.
Nivel 4 incorpora mediciones a la ejecución del proceso y así puede verificar su correcta ejecución.
Nivel 5 es capaz de identificar mejoras del proceso permitiendo así la mejora continua.

f) ¿Cuál es el propósito del calificador "friend"?
El calificador friend permite otorgar visibilidad a otras clases o funciones globales de todos los atributos privados o protegidos de una clase.

g) Mencione dos ventajas de crear archivos .h y .cpp independientes para cada clase de un proyecto respecto de usar un único archivo que incluya los encabezados y las implementaciones.
** Facilita la reutilización de código pues así puedo reutilizar clases específicas en otros proyectos.*
** Permite una compilación más eficiente pues ante un cambio es preciso recompilar sólo la parte que fue modificada.*

h) ¿Muestre la parte de la implementación de Persona que permite asignar el valor del atributo RUT a un objeto instancia de Persona?

```
class Persona {
  const string RUT;  …..}
```

En Persona.h Se debe tener un constructor para fijar el rut. Puede ser:
Persona(string _rut);
En Persona.cpp se debe poner su implementación: Persona::Persona(string _rut): RUT(_rut){}

Segunda Parte, con apuntes (68 minutos)

2.- (34 puntos) En la versión C++ de la traea1 se tiene segmentos de código para la clase Ball según se muestra en Ball.h y Ball.cpp. Muestre el contenido de CVector.h y CVector.cpp que permitan ejecutar las operaciones de instancias de CVector mostrados en los comentarios del método computeNextState.

Segmento de Ball.h

```
class Ball: public PhysicsElement {
private:
  static int id;  // Ball identification number
  const double mass;
  const double radius;
  CVector pos_t;     // current position at time t
  CVector pos_tPlusDelta;  // next position in delta time in future
  CVector speed_t;   // speed at time t
  CVector speed_tPlusDelta;   // speed in delta time in future
  Ball();
public:
  // .... un constructor y varios métodos .... //
  void computeNextState(double delta_t, MyWorld * world);
};
```

Segmento de Ball.cpp:

```
void Ball::computeNextState(double delta_t, MyWorld * world) {
 Ball * pb;
   if ((pb=world->findCollidingBall(this))!= NULL){
     double massFactor = 2*pb->getMass()/(mass+pb->getMass());
     CVector centersDifference = pos_t – pb->getPosition();                    /*  CVector operation */
     double dotProduct = (speed_t-pb->getSpeed()) % centersDifference ;  /*  CVector operation % para producto interno */
     speed_tPlusDelta= speed_t – massFactor*dotProduct*centersDifference/2.0;          /* CVector operations */
     pos_tPlusDelta = pos_t;
   } else {
     speed_tPlusDelta = speed_t;
     pos_tPlusDelta = pos_t + speed_t*delta_t;                                 /* CVector operations */
   }
}
```

CVector.h

```
class CVector {                        // 4 puntos por estructura general de la clase Nombre, miembros privados y públicos.
  private:
    double x, y;  // we will use cartesian coordinates
  public:
    /* otros constructores y métodos */
    CVector operator+(const CVector &v) const;        //   ...post + speed_t*delta_t                    3 puntos
    CVector operator-(const CVector &v) const;        // Cálculo de centersDifference                   3 puntos
    CVector operator*(double scalar) const;           // ...*dotProduct*CentersDifference...            3 puntos
    double operator%(const CVector &v) const;         // cálculo de dotProduct                          3 puntos
    CVector operator/(double scalar) const;           // .... centersDifference/2.0                     3 puntos
    friend CVector operator*(double scalar, const CVector &v);  // speed_t*delta_t                     3 puntos
};
```

CVector.cpp

```
CVector CVector::operator+(const CVector &v) const {                  // cada implementación 2 puntos
  CVector temp(x+v.x, y+v.y);
  return temp;
}
CVector CVector::operator-(const CVector &v) const {                  // 2 puntos
  CVector temp(x-v.x, y-v.y);
  return temp;
}
CVector CVector::operator*(double scalar) const {                     // 2 puntos
  CVector temp(scalar*x, scalar*y);
  return temp;
}
CVector CVector::operator/(double scalar) const {                     // 2 puntos
  CVector temp(x/scalar, y/scalar);
  return temp;
}
```

```
double CVector::operator%(const CVector &v) const {                    // 2 puntos
  return x*v.x + y*v.y;
}

CVector operator*(double scalar, const CVector &v){                    // 2 puntos
  CVector temp(scalar*v.x, scalar*v.y);
  return temp;
}
```

3.- (34 puntos) Sin hacer uso de algoritmos genéricos disponibles en la bliblioteca estándar de plantillas, se pide que usted implemente la función plantilla myRemove_if. Esta plantilla admite como parámetros un vector y un objeto que actuará como función a través del operador (). myRemove_if elimina (es decir borra o remueve) del arreglo todos los elementos cuya evaluación hecha por el objeto, que actúa como función, retorne verdadero.
Un ejemplo de uso de esta función sería:
vector<Student> curso;
MyClass objFunction; // objFunction alguna función para la cual tiene sentido objFunction(s) con s un estudiante.
…...
myRemove_if(curso, objFuction); // elimina de curso todos los estudiantes **s** para los cuales objFunction(s)
// retorne verdadero.

a) (18 puntos) Implemente una plantilla para myRemove_if.
b) (16 puntos) Implemente una clase MyFunction para que el siguiente código permita eliminar del vector v todos los enteros múltiplos de N.
int N=3;
MyFunction func(N);
vector<int> v;
….. // aquí se asignan muchos enteros al vector v
myRemove_if(v, func); // con esto se logra eliminar los múltiplos de 3, en este caso, del vector v

a)

```
#ifndef MY_REMOVE_IF
#define MY_REMOVE_IF
#include <vector>
using namespace std;

template <class T1, class T2>                                          // 2 puntos
void myRemove_if(vector<T1> &v, const T2 &func) {                      // por parámetros: 8 puntos
  typename vector<T1>::iterator iter=v.begin();       // si no pone typename es OK.  Por implementación 8 puntos
  while(iter!=v.end()) {
    if(func(*iter))
      iter=v.erase(iter);
    else
      iter++;
  }
}
#endif
```

b)

```
class myFunction {                      // declaración 10 puntos
public:
  myFunction (int n);
  bool operator()(int i) const;
private:
  int N;
};

myFunction::myFunction(int n):N(n){    // 3 puntos
}
bool myFunction::operator()(int i) const{  // 3 puntos
  return (i%N==0);
}
```

Un contexto de uso de esta función sería (solo por completitud, no se pide en el certamen):

```
#include "P3.h"
#include <iostream>

using namespace std;

int main ()
{
  vector<int> myvector;
  for (int i=1; i<=50; i++) myvector.push_back(i);

  cout << "Initially myvector contains:";

  for (vector<int>::iterator it = myvector.begin() ; it != myvector.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';

  myFunction func(3);
  myRemove_if(myvector,func);
  cout << "Finally myvector contains:";
  for (vector<int>::iterator it = myvector.begin() ; it != myvector.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';

 return 0;
}
```