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) En la clase adjunta, ¿Cómo se asigna un valor inicial al atributo a_name?

class A { public: static string a_name; …..};

En la implementación de la clase, archivo .cpp, incluir: string A::a_name= “Asignatura ELO329”;

b) Por qué se recomienda partir los archivos de encabezados con algo del tipo:

#ifndef MI_ENCABEZADO

#define MI_ENCEBEZADO

La idea es evitar incluir más de una vez cada archivo. La definición debe ser única dentro del proyecto de manera que una vez incluido un archivo esa definición bloquee repeticiones de ésta. Así el compilador no reportará errores de doble definiciones.

c) ¿Bajo qué condición y con qué fin se recomienda implementar un método dentro de la declaración de su clase (conocido como función miembro o método in-line)?

Se recomienda hacer esto cuando el código del método es pequeño. Esto se hace con el propósito de obtener un código más rápido debido a que en cada llamado a ese método el compilador pondrá el código del mismo en lugar de un llamado a éste. Con esto se logra mantener el pipeline de ejecución de los procesadores modernos.

d) Se tiene la función: void foo(A a);

Cuando ésta es invocada como en foo(mi_a), con **mi_a** instancia de la clase A, ¿Cómo **a** toma el valor de **mi_a**? Muestre un código equivalente para ese paso de valor si usted tuviera que crear la variable **a** y asignarle el valor.

Corresponde a la creación de la variable local a a partir del constructor copia. El código puesto por el compilador equivale a:

A a(mi_a);

e) ¿Qué problema tiene el siguiente código? ¿Sugiera una forma para superar el problema sin cambiar el diseño?

```
class Base {
 public:  Base();
   virtual ~Base() {}
 ...
};
```

```
class Derivada: public Base {
  public: Derivada (int a) {
            p = new int(a);
           }
    virtual ~ Derivada() {  //no es obligado
              delete p;
           }
....
  private: int * p;
};
```

```
int main () {
   Base *pB = new Derivada(3);
    // cómputos
   delete pB;
    // otras tareas
}
```

El problema es la creación de zonas de memoria no alcanzables por el código (fuga de memoria).

Para resolverlo se debe asegurar que al eliminar el objeto apuntado por el puntero a Base se invoque el destructor de objeto (no sólo el de Base). Esto se logra incluyendo el destructor en Base con calificador virtual.

En clase Base incluir bajo public: virtual ~Base(){}

En este caso se recomienda también incluir el calificador virtual en el destructor de Derivada.

f) Considere la clase Complejo mostrada ¿Qué código debe agregar para permitir operaciones como la indicada a la derecha?

class Complejo { // clase para números complejos.
public: double real, double img;
Complejo operator * (double f, const Complejo &z);
:
};
En implementación:
Complejo operator * (double f, const Complejo &z) {
Complejo r;
r.real=f*z.real; r.img=f*z.img;
return r;
}
	Se pide hacer posible código como aquel en negrita:
	Complejo z1;
	z1.real = z1.img = 3.5;
	Complejo **z2 = 3.0* z1**;

g) Explique qué caracteriza una organización con certificación CMM nivel 1 y otra de nivel 4.

CMM nivel 1: la organización no tiene definido procesos ni utiliza al experiencia previa, el resultado obtenido depende de la calidad de cada individuo.

CMM nivel 4: La empresa tiene procesos bien definidos y efectúa mediciones que le permiten controlar la calidad de sus resultados los cuales son predecibles.

h) ¿Qué es un patrón de diseño?

Un patrón de diseño es una solución general reusable a un problema común recurrente.

Segunda Parte, con apuntes (68 minutos)

2.- (34 puntos) Complete el diseño e implemente la clase Fecha para que el código mostrado funcione.
Nota: No se pide considerar día de la semana y suponga todos los meses de 30 días.

```
class Fecha {
public:
    Fecha();
    Fecha(unsign char dia, unsign char mes, unsign int year);
    :
};
int main (void) {
  Fecha inicio(9,6,2015);
  Fecha fin(1,9,2015);
  cout << "Duración " << (fin-inicio).totalDias() << " días" << endln; // muestra diferencia en días.
  Fecha plazo (0,0,6);
  cout << "Juan termina su carrera el " << inicio + plazo << endln;  // fecha sale como día/mes/año
  cout << "La extensión de su carrera se puede extender hasta el " << inicio + 1.5*plazo << endln;
}
```

```
#include <iostream>
#include <ostream>
using namespace std;

class Fecha {
public:
    Fecha();
    Fecha(unsigned char dia, unsigned char mes, unsigned int year);
    long totalDias() const;  // por primer cout <<
    Fecha operator- (const Fecha &f) const;  // por resta
    Fecha operator+ (const Fecha &f) const;  // por suma
    friend ostream & operator<< (ostream & os, const Fecha &f); // por cout << fecha
    friend Fecha operator* (double p, const Fecha &f);

private:
    Fecha(long d);
    long dias;   // los atributos los decide el diseñador.
};
```

```
Fecha::Fecha(){
        dias=0;
}
Fecha::Fecha(unsigned char dia, unsigned char mes, unsigned int year){
        dias = dia + mes*30+year*360;
}

Fecha::Fecha(long d){
        dias = d;
}

long Fecha::totalDias() const {
    return dias;
}

Fecha Fecha::operator -(const Fecha& f) const {
        return Fecha(this->dias-f.dias);
}

Fecha Fecha::operator +(const Fecha& f) const {
        return Fecha(this->dias+f.dias);
}

ostream & operator << (ostream &os, const Fecha &f) {
        int year = f.dias/360;  // 1 año = 12 meses de 30 días
        int dayOfYear= f.dias%360;
        unsigned int month = dayOfYear/30;
        unsigned int dayOfMonth = dayOfYear%30;
    os << dayOfMonth << "/" << month << "/" << year;
    return os;
}
Fecha operator* (double p, const Fecha &f) {
        return Fecha(p*f.dias);
}

int main (void) {
  Fecha inicio(9,6,2015);
  Fecha fin(1,9,2015);
  cout << "Duración " << (fin-inicio).totalDias() << " días"<< endl; // muestra diferencia en días.
  Fecha plazo (0,0,6);
  cout << "Juan termina su carrera el " << inicio + plazo << endl;  // fecha sale como día/mes/año
  cout << "La extensión de su carrera se puede extender hasta el " << inicio + 1.5*plazo << endl;
}
```

3.- (34 puntos) Se necesita crear una clase para representar cartas del naipe Inglés. Se propone un bosquejo de clase como se muestra a continuación.
a) Completar la clase Carta para que el código mostrado en main compile sin problemas.
b) Muestre una versión más general de la función maximo para que pueda trabajar sobre tipos de datos distintos a Carta.

```
#include <iostream>
#include <vector>
#include <stdlib.h>
#include <time.h>
using namespace std
const string Pinta[]={"Corazon", "Diamante", "Trebol", "Pica"};
const string Valor[]= {"dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve", "diez", "J", "Q", "R", "A"};

class Carta {
public:
  Carta();
  Carta(const Carta &c);

// Completar


private:
  const unsigned char pinta;
  const unsigned char valor;
};

Carta::Carta():pinta(0), valor(13)  {
}

const Carta& maximo(const vector<Carta> &c) {
  int max = 0;
  for (unsigned int i=1; i<c.size(); i++)
    if (c[i]> c[max])
      max=i;
  return c[max];
}

int main (void) {
  srandom(time(NULL)); // semilla números aleatorios
  vector<Carta> baraja;
  Carta c1(random()%13, random()%4);
  baraja.push_back(c1);
  for (int i=0; i< 4; i++)
    baraja.push_back(*(new Carta(random()%13, random()%4)));
  cout << maximo(cartas)<< endl;
  for (int i=0; i< 4; i++)
    baraja.push_back(*(new Carta(random()%13, random()%4)));
  const Carta &c=maximo(baraja);
  cout << c << endl;
}
```

```
#include <iostream>
#include <vector>
#include <stdlib.h>
#include <time.h>

using namespace std;

const string Pinta[]={"Corazon", "Diamante", "Trebol", "Pica"};
const string Valor[]= {"dos", "tres", "cuatro", "cinco", "seis", "siete", "ocho", "nueve", "diez", "J",
"Q", "R", "A"};
class Carta {
public:
  Carta();
  Carta(unsigned char v, unsigned char p);  // requerido por código main  2pts
  Carta( const Carta &c);

// Completar
  bool operator> (const Carta & c) const;  // requerido por maximo   4 pts
  const Carta & operator=(const Carta &c);  // requerido por vector pero no es obvio, no se exige
  friend ostream& operator<<(ostream &os, const Carta&d);  // requerido por cout <<   4 pts

private:
  const unsigned char pinta;
  const unsigned char valor;  // en esta implementación parte de cero
};

Carta::Carta():pinta(0), valor(12)  {   // 2 pts
}
```

```
Carta::Carta(unsigned char v, unsigned char p): pinta(p), valor(v) {   // 3 pts.
}
Carta::Carta(const Carta &c): pinta(c.pinta), valor(c.valor) {  // 3 pts.
}
bool Carta::operator> (const Carta& c) const {   // 3 pts
        return valor > c.valor;
}
const Carta & Carta::operator=(const Carta &c){
        return *this;
}

ostream & operator<<(ostream & os, const Carta&c){   // 3 pts.
  os << Valor[c.valor] << " de " << Pinta[c.pinta] << endl;
  return os;
}

template <class T>   // Parte b 10 pts.
const T& maximo(const vector<T> &c)
{  int max = 0;
  for (unsigned int i=1; i<c.size(); i++)
    if (c[i]> c[max])
      max=i;
  return c[max];
}

int main (void) {
  srandom(time(NULL)); // semilla números aleatorios
  vector<Carta> baraja;
  Carta c1(random()%13, random()%4);
  baraja.push_back(c1);
  for (int i=0; i< 4; i++)
    baraja.push_back(*(new Carta(random()%13, random()%4)));
  cout << maximo(baraja)<< endl;
  for (int i=0; i< 4; i++)
    baraja.push_back(*(new Carta(random()%13, random()%4)));
  const Carta &c=maximo(baraja);
  cout << c << endl;
}
```