Soluciones al examen de Lenguajes de Programación II
Convocatoria de junio de 2000

PREGUNTA 1

```
#include <iostream.h>

class  lista
{
  int *v;
  int tam;
  int estA (int);
  public:
  lista ( void ) { v=NULL; tam=0; }
  lista (lista & ); //Constructor copia
  ~lista ( ) { delete v;}
  lista operator+=( int );
  friend lista operator+ ( lista, lista );
  lista operator= ( lista  );
  friend lista operator- ( lista, int );
  friend int operator== ( lista, lista );
  friend ostream& operator<< (ostream&, lista );
  friend istream& operator>> (istream&, lista&);
};

int lista::estA(int val)
{
  for (int i=0; i<tam; i++)
     if (v[i]==val) return 1;
   return 0;
}

lista::lista ( lista &l )
{
  v=new int [l.tam];
   tam=l.tam;
   for (int i=0; i<tam; i++)
     v[i]=l.v[i];
}

lista lista::operator+= ( int val )
{
  int *vaux;

   vaux=new int [tam+1];
   for ( int i=0; i<tam; i++)
     vaux[i]=v[i];
   vaux[tam]=val;
   tam++;
   delete v;
   v=vaux;
  return *this;
}

lista operator+ (lista l1, lista l2 )
{
  for ( int i=0; i<l1.tam; i++)
```

```
      if ( !l2.estA(l1.v[i]) ) l2+=l1.v[i];
   return l2; //Aunque modifiquemos l2, no pasa nada porque la
hemos pasado
                      // por valor.
}

lista lista::operator= ( lista l )
{
   if (tam!=0) delete [] v;
   v=new int [l.tam];
   tam=l.tam;
   for (int i=0; i<tam; i++)
      v[i]=l.v[i];
   return *this;
}

ostream& operator<< (ostream& os, lista l )
{
  for (int i=0; i<l.tam; i++)
      os<<l.v[i]<<",";
   os << endl;
   return os;
}

int operator== ( lista l1, lista l2 )
{
  if ( l1.tam!=l2.tam) return 0;
  for (int i=0; i<l1.tam; i++)
      if (!l2.estA(l1.v[i])) return 0;
   return 1;
}

lista operator- ( lista l, int val )
// Como pasamos l por valor, las modificaciones no perduraran
{
  if ( !l.estA(val) ) return l;
   int *vaux = new int [l.tam-1];
   for ( int i=0, j=0; i<l.tam; i++)
      if ( l.v[i]!=val)
                 vaux[j++]=l.v[i];
   delete l.v;
   l.tam--;
   l.v=vaux;
   return l;
}

istream& operator>> (istream& is, lista &l)
{
  int val=0;

   while (val>=0)
   {
      cout << "Introduzca un valor (negativo para finalizar)";
       is >> val;
       if (val>=0)
            l+=val;
   }
   return is;
}
```

```
void funcion ( lista v1 )
{
  lista v2, v3;

  v2+=1; //Inserta 1 en v2
  v3=v2+v1; //Junta v2 y v1
   cout << v3;
  cout << "Inserte una nueva lista de enteros:";
  cin >> v2;
  v3=v2-5; //Elimina el cinco de la lista v2
  if ( v3==v2) cout<<"La lista no contenía ningún 5";
}
```

PREGUNTA 2

```
#include <iostream.h>
#include <math.h>

class fruta
{
  protected:
  char variedad[30];
   float diametro;
   fruta *next;
  public:
   fruta (void) {next=NULL;}
   virtual float precio ( void )=0;
   friend class BolsaDeFruta;
};


class Naranja: public fruta
{
  float acidez;
   public:
   float precio ( void ) { return fabs(10.0*diametro - acidez);}
   friend class BolsaDeFruta;
};

class Mandarina: public fruta
{
  int pepitas;
   public:
   float precio ( void ) { return fabs(10.0*diametro -
((pepitas)?25:0));}
   friend class BolsaDeFruta;
};

class BolsaDeFruta
{
  fruta *P;
   public:
   BolsaDeFruta ( void ) { P=NULL; }
   ~BolsaDeFruta ( ) { VaciarCesta ();}
   void VaciarCesta ( void );
   void InsertarFruta ( void );
   float Precio ( void );
};

void BolsaDeFruta::VaciarCesta (void)
{
  fruta *aux;

   while (P!=NULL )
   {
      aux=P;
       P=P->next;
       delete aux;
   }
}

void BolsaDeFruta::InsertarFruta ( void )
```

```
{
  fruta *aux;
   int tipo;

   do
   {
      cout << "¿Qué desea insertar (Naranja=1, Mandarina=2, 
Salir=0";
      cin>>tipo;
      if (tipo==1)
      {
         Naranja *Naux=new Naranja;
         cout << "Acidez:";
         cin >> Naux->acidez;
         aux=Naux;
      }
      if (tipo==2)
      {
         Mandarina *Maux=new Mandarina;
         cout << "¿Pepitas=1, Sin Pepitas=0";
         cin>>Maux->pepitas;
         aux=Maux;
      }
      if (tipo!=0)
      {
         cout << "variedad:"; cin >> aux->variedad;
         cout << "diametro:"; cin >> aux->diametro;
         aux->next=P;
         P=aux;
      }
   }
   while ( tipo!=0 );
}

float BolsaDeFruta::Precio ( void )
{
  fruta *aux;
   float total=0.0;

   for ( aux=P; aux!=NULL; aux=aux->next )
     total+=aux->precio();
   return total;
}
```

PREGUNTA 3.

Responde brevemente a las siguientes preguntas:

a) ¿Qué son variables de clase e instancia? ¿Qué son funciones de clase?

Las variables de instancia se definen en una clase, pero pueden tomar valores diferentes para cada objeto de esa clase. Por su parte, las variables de clase son comunes a toda la clase, por lo que su contenido es el mismo para todos los objetos de la clase.

Las funciones de clase actúan sobre la clase y no sobre un objeto en concreto. No pueden actuar sobre *this, y en caso de que actúen sobre algún objeto de la clase en que están definidas, éste o bien será declarado como local, o será un argumento de la lista de parámetros que admite la función.

b) ¿Para qué se usa la palabra **friend** en C++?
   Permite acceder a la parte protegida de una clase a funciones que no pertenecen a esa clase.
c) ¿Cuándo se define una herencia como virtual?
   Cuando tenemos una clase A de la que derivan varias clases B1, B2, B3, ..Bn y además definimos una nueva clase C que deriva de más de una de estas clases Bx. En ese caso debe definirse la herencia como virtual para no heredar varias veces lo que se definió en A.
d) ¿Cuáles son las tres principales palabras clave del manejo de excepciones en C++?
   try, catch y throw.
e) ¿Cómo se llama la clase de E/S que permite abrir ficheros para entrada y salida?
   fstream
f) ¿Se puede sobrecargar una función genérica? Pon un ejemplo.
   Si, pues no dejan de ser funciones y por lo tanto también son sobrecargables.

```
template <class T>
void f ( T t, int i) { ..}
template <class T>
void f ( T t, int i, float f) { ..}
```

¿Qué es **this**? ¿Es útil la palabra **this** fuera de las funciones de una clase?
Dada una función de una clase, this es la dirección del objeto sobre el que se ejecuta una función.
No tiene sentido fuera de la clase.
Cuando una herencia se define como **protected**, ¿cómo cambia el nivel de acceso de los miembros de la clase base?
Public → Protected
Protected → Protected
Private → Private
`