FUNDAMENTOS DE PROGRAMACIÓN
1º INGENIERÍA INFORMÁTICA (Plan 2000)
6 de febrero de 2001

NORMAS DE EXAMEN: Notas inferiores a 3.5 en este ejercicio implican suspender la asignatura (en primera convocatoria). Las notas superiores a 3.5 promediarán con la calificación del proyecto de laboratorio según la siguiente proporción: 40% nota del examen, 60% nota del proyecto.

1.- Explicar qué hace la siguiente función (no se pide la traducción en lenguaje natural de cada línea de código, sino su finalidad última). 1 pto.

```
bool misterio (string a, string b)
{
   unsigned int i;
   bool res;

   res = true;

   if ( a.length() == b.length() )
   {
      i = 0;
      while ( i < a.length() )
         if ( a[i] == b[i] ) i++;
         else
         {
            res = false;
            i = a.length();
         }
   }
   else res = false;

   return res;
}
```

2.- Detectar y explicar cómo se pueden corregir los errores de compilación que existan en las siguientes funciones (pueden haber 0 o más errores en cada función): 2 ptos.

a)

```
/*1*/   void f ()
/*2*/   {
/*3*/      cout << "Funcion f" << endl;
/*4*/      int g()
/*5*/      {
/*6*/         return 1;
/*7*/      }
/*8*/      return g;
/*9*/   }
```

b)

```
/*1*/   int suma (int x, int* y)
/*2*/   { return x + y; }
```

c)

```
/*1*/   void f (float a);
```

```
/*2*/   {
/*3*/      float a;
/*4*/      cin << a << endl;
/*5*/   }
```

d)

```
/*1*/   void f ( float y )
/*2*/   {
/*3*/      switch (y)
/*4*/      {
/*5*/         case 1.0: cout << "uno" << endl; break;
/*6*/         case 1.5: cout << "uno y medio" << endl; break;
/*7*/         case 2.0: cout << "dos" << endl; break;
/*8*/         default: cout << "Mayor que dos" << endl; break;
/*9*/      }
/*10*/  }
```

3.- Escribir una función recursiva que permita invertir las posiciones de todos los elementos de un vector de enteros. De manera que el primer elemento del vector pase a ser el último (y viceversa), el segundo el penúltimo y así, sucesivamente. Para ello, se supone la existencia de la siguiente definición de tipo: `typedef int vector[MAX];` 2 ptos.

4.- Escribir una función que calcule el valor máximo, el valor mínimo y el valor medio de un conjunto de números reales almacenados en un archivo en disco, con los siguientes requisitos: 3.5 ptos.

(a) La función debe de escribir los tres datos estadísticos que calcula en un archivo en disco.

(b) El archivo de salida se llamará igual que el archivo original pero substituyendo su extensión por `".std"` (se supone que el archivo original siempre tendrá una extensión de 3 caracteres). Por ejemplo, si el archivo original se llama *"experimento1.dat"*, el archivo con la estadística se llamará *"experimento1.std"*.

(c) La función no puede utilizar ningún array para almacenar temporalmente información.

(d) Se permite escribir más de una función para modularizar mejor el código.

(e) La función a escribir debe de cumplir el siguiente prototipo:

```
bool estadistica ( string datos );
```

Siendo *datos* el nombre del archivo que se desea analizar y el valor de retorno de la función indicará si se ha producido (o no) algún error al manipular los archivos.

5.- Escribir la(s) declaración(es) de tipos necesaria(s) para especificar un tablero de ajedrez (cuadrado de 8x8 casillas) para una aplicación que requiere conocer el color de cada casilla, si está ocupada o no y qué pieza la ocupa. 1.5 ptos.