LENGUAJES DE PROGRAMACIÓN

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (3 puntos)

Indique la veracidad o falsedad de cada una de las afirmaciones siguientes, explicando detalladamente en cada caso el motivo de su respuesta.

A. (0.5 puntos) En la máquina de Von Neumann la unidad aritmética recibe las señales de control de la unidad de entrada/salida.

B. (0.5 puntos) Desde el punto de vista de los valores finalmente almacenados en las variables `x` e `i`, ejecutar la sentencia:

```
x = ++i;
```

es equivalente a ejecutar las dos sentencias siguientes:

```
x = i;
i = i+1;
```

C. (0.5 puntos) Para concatenar literales *string* en C++ puede emplearse el operador +. Por ejemplo, la siguiente sentencia es correcta:

```
std::string ab = "a"+"b";
```

D. (0.5 puntos) Fortran 90 tiene un bucle lógico postcondición **repeat-until**.

E. (0.5 puntos) Cada elemento de una lista doblemente enlazada tiene un único puntero.

F. (0.5 puntos) El algoritmo de la STL

```
replace_copy(pBegin, pEnd, pResultBegin, val, val1)
```

copia la secuencia origen en la secuencia destino, eliminando de la copia los elementos cuyo valor sea `val` o `val1`.

Pregunta 2 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <iomanip>

int main () {
   double* p;
   double a[2];
   double x, y, z;
   a[0] = 0.123456;
   a[1] = 8.6789;
   x = 1;
   y = 2;
   z = 3;
   p = a;
   x = p[1];
   std::cout << std::scientific << std::setprecision(2) << x <<std::endl;
   std::cout << y << std::endl;
   std::cout << z << std::endl;
   {
     double x = 40;
     a[1] = x;
     y = x;
     z += y;
   }
   std::cout << p[1] << std::endl;
   std::cout << std::setprecision(3) << a[1] <<std::endl;
   std::cout << x << std::endl;
   std::cout << y << std::endl;
   std::cout << z << std::endl;
   p = p + 1;
   std::cout << p[0] << std::endl;
   std::cout << a[0] << std::endl;
   return 0;
 }
```

Pregunta 3 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <map>
#include <string>

int main()
{
   std::map<std::string, int> mapa;
   mapa["A"] = 10;
   mapa["B"] = 20;
   mapa["C"] = 30;
   mapa["D"] = 40;
   std::cout << mapa.count("A")  << std::endl;
   std::cout << mapa.count("AB") << std::endl;
   std::map<std::string, int>::iterator p = mapa.find("B");
   while ( p!=mapa.end() ) {
      std::cout << (*p).first << " " << (*p).second << std::endl;
      p++;
   }
   mapa.erase("B");
   p = mapa.begin();
   while(p!=mapa.end()){
      std::cout << (*p).first << " " << (*p).second << std::endl;
      p++;
   }
   return 0;
 }
```

Pregunta 4 (2 puntos)

Se desea introducir en un sintetizador musical la secuencia de frecuencias correspondiente a una obra musical. Como ayuda para ello, se propone realizar un programa que traduzca una obra musical escrita en la notación clásica a la secuencia equivalente de frecuencias.

La notación clásica está compuesta por 12 notas, que en orden creciente de frecuencia son:

C, Cs, D, Ds, E, F, Fs, G, Gs, A, As, B

La relación entre la frecuencia de una nota y la frecuencia de la nota consecutiva en la escala es $\sqrt[12]{2}$. Es decir:

$$f(N_{i+1}) = \sqrt[12]{2} \cdot f(N_i)$$

donde N_{i+1} representa la nota consecutiva a la nota N_i, y $f(N_{i+1})$ y $f(N_i)$ representan las frecuencias de dichas notas.

La frecuencia de la nota A es $f(A) = 440$Hz.

Escriba un programa en C++ que realice las acciones siguientes:

1. Solicitar por consola al usuario que éste introduzca por consola la secuencia de caracteres correspondiente a la serie de notas musicales de la obra en la notación clásica. Almacenar dicha secuencia en un *string* llamado *obra*.
2. Calcular la equivalencia entre las 12 notas musicales de la escala y sus frecuencias, almacenando dichas frecuencias en un array llamado *notas*.
3. Empleando sentencias **for** y **switch**, ir asignando a cada nota de la obra su frecuencia correspondiente, almacenando consecutivamente las frecuencias de la obra en un vector de **double** llamado *freqs*.
4. Mostrar en consola la secuencia de frecuencias de la obra que se encuentran almacenadas en el vector *freqs*.
5. Finalizar.

Pregunta 5 (3 puntos)

Escriba en C++ una función que calcule el número de elementos de un vector que se encuentran dentro de un cierto intervalo, y un programa que lea números reales de un fichero de texto y muestre en la consola cuántos de estos datos están contenidos en un cierto intervalo, valiéndose para ello de la función anteriormente definida. A continuación se explica todo ello con detalle.

5.a (1.5 puntos) Defina una función en C++ que acepte tres parámetros: una referencia a un vector de elementos **double** y dos variables **double**. Se muestra a continuación el prototipo de la función, la cual puede lanzar fuera de sí una excepción:

```
int numDatosEnIntervalo(std::vector<double>&, double, double)
       throw (std::invalid_argument);
```

La función, que devuelve un valor **int**, debe realizar las acciones siguientes:

1. *Comprobar que el intervalo esté bien definido*. Los dos argumentos **double** son el extremo inferior y superior del intervalo, respectivamente. Debe satisfacerse que el extremo inferior sea menor o igual al extremo superior. Si no se satisface esta condición, la función debe lanzar una excepción.
2. *Contar los elementos del vector que están en el intervalo*. Para ello, debe examinar uno a uno los elementos del vector referenciado. Un elemento está dentro del intervalo si su valor es mayor o igual al extremo inferior del intervalo y además es menor o igual al extremo superior del intervalo.
3. *Devolver el resultado de la cuenta anterior*.

5.b (1.5 puntos) Escriba un programa en C++ que realice las acciones siguientes.

1. *Apertura para lectura de un fichero de texto*. El nombre del fichero de texto en el cual se encuentran los datos debe almacenarse en una constante global del tipo `std::string`, llamada *nombreFich*, cuyo valor debe ser `"datos.txt"`. El programa debe abrir el fichero de texto para lectura. Si se produce error, debe mostrar un mensaje en la consola indicándolo y terminar.
2. Declarar un vector de elementos **double** llamado `vectorDatos`.
3. *Lectura del fichero de texto y escritura en el vector*. El fichero de texto contiene datos, que son números reales. El programa debe ir leyéndolos y añadiéndolos al final del vector `vectorDatos`.

4. *Entrada por consola de los extremos del intervalo*. El programa debe solicitar al usuario que éste introduzca por consola los extremos inferior y superior del intervalo. Ambos valores deben almacenarse en variables **double** llamadas `intervL` y `intervH`, respectivamente.
5. Debe invocar la función anteriormente definida, pasando como argumento la referencia al vector y los valores de los extremos del intervalo, y debe escribir en la consola el valor devuelto por la función.

   El programa debe estar preparado para capturar y tratar la excepción que puede ser lanzada por la función. Si se captura una excepción, el programa debe mostrar un mensaje en la consola indicándolo.
6. Terminar.