LENGUAJES DE PROGRAMACIÓN

INSTRUCCIONES
Por favor, entregue esta primera hoja 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 el lenguaje ALGOL 58, las variables vectoriales pueden tener un máximo de tres dimensiones.

B. (0.5 puntos) Un preprocesador se encarga de procesar un programa inmediatamente después de que éste sea compilado.

C. (0.5 puntos) Los operandos de las expresiones relacionales sólo pueden ser de tipo Booleano.

D. (0.5 puntos) La variable `var` declarada en la sentencia de C++

```
std::list<int>::iterator var;
```

es una lista doblemente enlazada.

E. (0.5 puntos) En el lenguaje C++ existe el operador `?`, que es un operador ternario que en algunos casos puede usarse como alternativa al operador `if`.

F. (0.5 puntos) Dentro de una función en C++ no puede definirse otra función.

Pregunta 2 (1 punto)

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

```
#include <iostream>
#include <vector>
#include <algorithm>

void writev(std::vector<int>& v) {
   for (unsigned int i = 0; i < v.size() - 1; i++)
      std::cout << v[i] << ",";
   std::cout << v[v.size() - 1] << std::endl;
   return;
}

int main() {
   std::vector<int> v;
   for (int i = 0; i < 10; i++)
      v.push_back(2 * i);
   writev(v);
   std::vector<int>::iterator pBegin = v.begin() + 1;
   std::vector<int>::iterator pEnd = v.begin() + 4;
   std::cout << "(*pBegin): " << (*pBegin)
             << "\t(*pEnd): " << (*pEnd) << std::endl;
   v.push_back(0);
   writev(v);
   reverse(pBegin, pEnd);
   writev(v);
   std::cout << "(*pBegin): " << (*pBegin)
             << "\t(*pEnd): " << (*pEnd) << std::endl;
   for (int val = 0; val < 3; val++)
      std::cout << count(v.begin() , v.end(), val)
                << 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.insert(std::pair<std::string, int>("Mercurio", 1));
   mapa.insert(std::pair<std::string, int>("Venus", 20));
   mapa["Tierra"] = 3;
   mapa["Marte"] = 4;
   mapa.insert(std::pair<std::string, int>("Mercurio", 5));
   std::map<std::string, int>::iterator it = mapa.begin();
   while ( it != mapa.end() ) {
      std::cout << it->first  << " :: "
                << it->second << " ";
      it++;
   }
   std::cout << std::endl;
   mapa.erase(--it);
   mapa["Venus"] = 10;
   mapa.insert(std::pair<std::string, int>("Jupiter", 9));
   it = mapa.find("Mercurio");
   while ( it != mapa.end() ) {
      std::cout << it->first  << " :: "
                << it->second << " ";
      it++;
   }
   std::cout << mapa.count("Mercurio") << std::endl;
   return 0;
}
```

Pregunta 4 (2.5 puntos)

Se desea calcular la masa de una molécula a partir de su composición química y siendo conocidas las masas de los átomos que la componen. Este último dato se obtiene de un fichero de texto llamado *masas.txt*, el cual contiene en la primera columna los símbolos de los elementos químicos y en la segunda columna sus respectivas masas atómicas. A continuación de muestran las primeras filas del fichero *masas.txt*, donde puede observarse que los elementos no están ordenados.

```
H 1.008
C 12.011
O 16.000
Na 22.990
Cl 35.450
```

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

1. Mediante un mensaje escrito en la consola, solicitar al usuario que escriba en la consola la fórmula molecular.
2. Leer la fórmula introducida por consola y almacenarla en un *string* llamado `molecula`.
3. Comprobar si la fórmula introducida es válida, para lo cual debe leer el fichero *masas.txt*.

   La fórmula debe ser la concatenación sucesiva de símbolos de elementos químicos (tal como se encuentran en la primera columna del fichero *masas.txt*) y números enteros mayores que cero. Estos últimos indican el número de átomos de cada elemento presentes en la molécula. Veamos algunos ejemplos:

   - La fórmula de la molécula de agua es `H2O1`, dado que dicha molécula está compuesta por dos átomos de hidrógeno (`H`) y un átomo de oxígeno (`O`).
   - La fórmula de la molécula de benceno es `C6H6`, dado que tiene seis átomos de carbono (`C`) y seis átomos de hidrógeno.
   - La fórmula de la molécula de la sal común (cloruro de sodio) es `Na1Cl1`, dado que está compuesta por un átomo de sodio (`Na`) y un átomo de cloro (`Cl`).
4. Si la fórmula no es válida, el programa debe indicarlo mediante un mensaje escrito en la consola y terminar.
5. Si la fórmula es válida, el programa debe calcular y escribir en la consola la masa de la molécula, en formato fijo con tres dígitos decimales. Por ejemplo, la masa de la molécula de agua (`H2O1`) es: $1.008 \cdot 2 + 16.000 \cdot 1 = 18.016$.
6. Terminar.

Pregunta 5 (2.5 puntos)

El programa informático para la gestión de un almacén hace uso de las siguientes estructuras en C++:

```
struct Contenedor {
   std::string  itemID;
   int          unidades;
   int          posicion;
};

struct Pedido {
   std::string  itemID;
   std::string  clienteID;
   int          unidades;
};
```

Cada tipo de objeto almacenado tiene un identificador unívoco, que es el miembro `itemID` de las dos estructuras anteriores. Los objetos están empaquetados en contenedores. Cada contenedor contiene un determinado número de objetos del mismo tipo. Cada contenedor ocupa una posición en el almacén, que es identificada de manera unívoca mediante un número entero. Cada cliente tiene asignado un identificador que lo distingue de manera unívoca. Un pedido es una orden de compra que realiza un cliente, mediante la cual solicita que se le entregue un determinado número de unidades de un determinado producto.

Mediante la estructura `Contenedor` se asocian el identificador del tipo de objeto (`itemID`) y el número de unidades de dicho objeto (`unidades`) que se encuentran dentro de un contenedor ubicado en una posición del almacén (`posicion`).

Mediante la estructura `Pedido` se asocia el identificador del tipo de objeto (`itemID`) y el número de unidades del objeto (`unidades`) pedidas por el cliente (`clienteID`).

5.1 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
int getInventario(std::list<Contenedor> &lContenedores,
                  std::string           itemID);
```

que devuelva el número total de objetos que hay en el almacén del tipo indicado por el segundo argumento de la función (`itemID`). El primer argumento de la función es la lista de todos los contenedores que se encuentran en el almacén.

5.2 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
int getDemanda(std::vector<Pedido> &vPedidos,
               std::string         itemID);
```

que devuelva el número total de objetos pedidos del tipo indicado por el segundo argumento de la función (`itemID`). El primer argumento de la función es un vector con todos los pedidos.

5.3 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
int getBalance(std::list<Contenedor> &lContenedores,
               std::vector<Pedido>   &vPedidos,
               std::string           itemID);
```

que devuelva, para el tipo de objeto indicado por el tercer argumento de la función (`itemID`), la diferencia entre el número de objetos de ese tipo que hay en el almacén y el número de objetos de ese tipo pedidos. Para ello, invoque las funciones definidas en los Apartados 5.1 y 5.2.

5.4 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
std::map<std::string,int> getBalanceCompleto(
                     std::list<Contenedor> &lContenedores,
                     std::vector<Pedido>   &vPedidos);
```

que devuelva un mapa cuya clave es el `itemID` y cuyo valor es el número de unidades en el almacén menos el número de unidades pedidas de dicho objeto. En el mapa deben encontrarse todos los tipos de objeto presentes tanto en la lista de contenedores como en el vector de pedidos.

5.5 (0.5 puntos) Escriba una función en C++ con el prototipo siguiente:

```
bool logDesabastecimiento(
                   std::list<Contenedor> lContenedores,
                   std::vector<Pedido>   vPedidos,
                   std::string           nombreFichero);
```

que escriba en un fichero de texto cuyo nombre viene determinado por el tercer argumento de la función, borrando el contenido previo, lo siguiente:

- Los identificadores (`itemID`) de los objetos que satisfacen que el número de unidades existentes en el almacén es menor o igual al número de unidades pedidas.

- Para cada uno de los objetos anteriores, la diferencia entre las unidades almacenadas y las pedidas.
- Para cada uno de los objetos anteriores, los identificadores (`clienteID`) de todos los clientes que han pedido dicho objeto.

Toda la información de cada tipo de objeto debe escribirse en una línea del fichero, separando los datos por un espacio en blanco.

La función devuelve el valor *true* si la escritura del fichero se realiza correctamente y *false* en caso contrario (si se produce error al abrir el fichero para escritura).