SIMULACRO EXAMEN ENERO 2021

1.- (test1) La clase **Product** tiene dos campos de tipo *string* "*name*" y "*model*", y un campo de tipo entero "price". Codifica el fichero computer.h y computer.cc para la clase C*omputer* con la siguiente funcionalidad:

- Constructor con 3 parámetros. El primero es obligatorio y será *name*. *model* será el segundo parámetro con valor por defecto "Model". Y *price* será el tercer parámetro con valor por defecto igual 1. Si *name* llega como la cadena vacía tendrá que iniciarse a la cadena "ERROR". Si *model* llega como una cadena vacía, deberán ponerse a su valor por defecto antes mencionado. Si el precio llega como parámetro con un valor inferior a 1, debe ponerse a 1.
- Modificador setName() que recibe como parámetro el nuevo nombre. Si el nuevo nombre es la cadena vacía, no se hará el cambio y se devolverá *false*. En caso contrario se hará el cambio y se devolverá *true*.
- Modificador setModel() que recibe como parámetro el nuevo modelo. Si el nuevo modelo es la cadena vacía, no se hará el cambio y se devolverá *false*. En caso contrario se hará el cambio y se devolverá *true*.
- Modificador setPrice() que recibe como parámetro el nuevo precio que debe ser mayor que cero, en cuyo caso se devuelve *true*. Si es menor o igual que cero no se asigna y se devuelve *false*.
- Observadores getPrice(), getModel() y getName().
- Observador getString() que devuelve una cadena con name, seguido de una coma y un espacio, seguido de model, seguido de una coma y un espacio, seguido de price. Ejemplo: si name="PC1" y model="PM1" y price=250, entones devolverá: "PC1, PM1, 250" (**NOTA:** usa la función std::to_string() para convertir datos numéricos a string).

PUNTUACIÓN: 2 puntos

2.- (test2) La clase **Cart** representa una cesta de la compra de una tienda online y tiene un campo de tipo entero "id" y una lista de objetos de la clase *Product* del ejercicio anterior (para la lista **usar el tipo list de la STL** de C++). Codifica los ficheros cart.h y cart.cc para la clase *Cart* con la siguiente funcionalidad:

a) Constructor que recibe como parámetro obligatorio el *id* de la compra.
b) Observador getId().
c) Observador getN() que devuelve el tamaño de la lista.
d) El método addProduct() de tipo void que recibe un objeto de tipo *Product* y lo añade a la lista.
e) El método deleteProduct() que recibe dos parámetros. El primero es un string con un nombre y el segundo es otro string con un modelo. El método debe borrar el elemento de la lista cuyo nombre y modelo sean iguales a los pasados como parámetro. Si no existe ningún elemento con esos datos, no hace nada y devuelve *false*. Cuando lo encuentra, lo borra y devuelve *true*.
f) El método print(), de tipo void, que muestra en pantalla un producto por línea con su posición en la lista seguida de "/", seguido del tamaño de la lista, seguido de una coma, seguido de su nombre, una coma, modelo, una coma y finalmente el precio. Como en este ejemplo:

1/tamaño_lista,nombre,modelo,precio
2/tamaño_lista,nombre,modelo,precio
3/tamaño_lista,nombre,modelo,precio
…

Durante la ejecución del test2 deberá mostrarse en pantalla:

1/3,PC1,M1,111
2/3,PC2,M2,222
3/3,PC3,M3,333

g) El método write(), de tipo void, que escriba lo mismo pero en un fichero texto que se llamará "salida.txt". Después de la ejecución del test2 el contenido del fichero "salida.txt" será:

1/3,PC1,M1,111
2/3,PC2,M2,222
3/3,PC3,M3,333

OJO: no escribas ningún espacio en blanco en el fichero "salida.txt"

PUNTUACIÓN: 2 puntos

3.- (test3) La clase **Computer** deriva de la clase Product del ejercicio anterior y además gestiona una variable de tipo *int* denominada "id". Escribe el código de la clase *Computer* en los ficheros computer.h y computer.cc con la siguiente funcionalidad:

- Constructor que recibe como parámetro obligatorio el valor para "id", y a continuación, en este orden, name, model y price de la clase base también todos obligatorios. Si el id recibido es menor o igual a cero se asignará como id el valor 999.
- Observador getId().
- Modificador setId() que no permita valores menores o iguales que cero. Si el parámetro es menor o igual a 0 debe devolver *false* y no asignarlo. Si por el contrario es positivo debe asignarlo y devolver *true*.
- Observador getInfo() que devuelve una cadena formada por la concatenación de la información del Computer. Por ejemplo si el objeto tiene estos datos: id=66, name="PC1", model="PCM1", price 250; esta función debe devolver la cadena: "Computer 66 name PC1 model PCM1 price 250". Observar que no hay ninguna coma (**NOTA:** usa la función std::to_string() para convertir datos numéricos a string).

PUNTUACIÓN: 2 puntos

4.- (test4) Codifica una función que se llame `mysort()` y que reciba como parámetro un vector de enteros de la STL y lo ordene con la función sort() de la STL. El vector quedará ordenado tras la llamada a la función. Escribe todo el código exclusivamente en el fichero `mysort.cc` (en este ejercicio no es necesario hacer el fichero `mysort.h`). Al ejecutar el test4 verás que en cada uno de los 2 test se muestra un vector de enteros antes y después de llamar a tu función. Comprueba el fichero `test4.cc` si tienes alguna duda.

PUNTUACIÓN: 2 puntos

5.- (test5) La clase **Pareja** tiene una declaración de la siguiente forma:

```
class Pareja{
  private:
    double a_, b_;
};
```

Codifica en los ficheros `pareja.h` y `pareja.cc` los siguientes métodos para esta clase:

a) Constructor con los valores iniciales de ***a*** y ***b*** como parámetros obligatorios.
b) Observadores getA() y getB().
c) Sobrecarga los operadores + y – (suma o resta por componentes como si fuesen componentes de un vector o unas coordenadas).
d) Sobrecarga del operador >> (extractor) que saque en pantalla los valores de a y b entre corchetes. Ejemplo: [2.3,5.7]
e) Sobrecarga del operador << (insertador) que pida por teclado a y b de la siguiente forma (saldrá en pantalla en la ejecución del test5):
   Introduce a: *(el usuario la introduce a por teclado)*
   Introduce b: *(el usuario la introduce b por teclado)*

NOTA: En la ejecución del test5 se usarán todos los operadores.

PUNTUACIÓN: 2 puntos