UNIVERSIDAD NEBRIJA

ALUMNO:

EPS -Ingeniería Informática

Asignatura: G0460009 Programación II

Curso: 2022/2023 Examen: Final Fecha: 5-06-2023 - **Turno 1 (8:00 h)**
Semestre: 2º Convocatoria: Ordinaria

Parte Práctica (10 puntos; 70% nota final)
Tiempo: 2 horas y 40 minutos

Los ficheros correspondientes a los ejercicios se deben entregar en la actividad Examen Final a través del campus antes de la finalización del tiempo establecido.
Cada fichero se llamará EjercicioX, donde la X será el número de dicho ejercicio. Los únicos formatos válidos serán **txt** o **cpp** o **.h**, **sin comprimir**.

Criterios generales de evaluación

Funciones/Métodos: Si no se usa el paso por referencia constante cuando las variables de los parámetros de entrada no son de tipo simple.	40%
Tipos de datos y variables: • Uso de variables globales (fuera del ámbito de una función). • Si no se usan los tipos contenedor vistos en clase (std::array; std::vector; std::set; std::string, etc.) para las variables que lo necesiten.	 0% 0%
El **programa no compila** o **no se asemeja** a lo pedido**.**	0%
Si no se cumplen los **criterios de entrega** indicados en la **actividad/examen.**	0%

Criterios particulares de evaluación

El elemento evaluable no compila o no se asemeja a lo que se pide	0%
El elemento evaluable no se aproxima suficientemente a lo pedido	40%
El elemento evaluable se aproxima suficientemente a lo pedido	60%
El elemento evaluable funciona correctamente y las estrategias y elementos de código elegidos son adecuados.	100%

UNIVERSIDAD NEBRIJA

IMPORTANTE:

- Todos los ejercicios del examen deberán ser resueltos de forma **algorítmica**, es decir, la **solución** propuesta tendrá que ser **general** y **no particular** para unos determinados datos/valores.

- Todos los ejercicios resueltos sin utilizar funciones cuando sea apropiado se valorarán con una nota máxima del 60% sobre la calificación prevista.

- Se recomienda una primera lectura del examen completo para planificar la realización del examen. Y una segunda lectura detallada antes de la realización de cada uno de los ejercicios propuestos.

UNIVERSIDAD NEBRIJA

Turno 1: **8:20-11:00**

Ejercicio Nº1 (3 puntos)

Escriba un programa en C++11, ***Ejercicio1***, que permita probar el funcionamiento de la clase Hora de etiqueta, **Chora**, según las especificaciones que se describen a continuación.

Implementar la clase **Chora** que encapsule los siguientes miembros:

- Atributos: permite almacenar **dos valores enteros** (hora y minutos).

  Además, antes de guardar estos datos se tienen que validar tanto el valor de la hora como el valor de los minutos según lo siguiente: **(0,5 puntos)**

  - Si el valor de la hora no está entre 0 y 23 (ambos valores inclusive), se lanza una excepción para indicarlo.
  - Si el valor de los minutos no está entre 0 y 59, se lanza una excepción para indicarlo.

- Métodos:

  - Constructores **por defecto**, **paramétrico** y **por copia**, con las excepciones especificadas. **(0,75 puntos)**
  - Métodos **getter** y **seter** para las variables atributos encapsuladas. **(0,5 puntos)**
  - El **operador ">>"** sobrecargado para poder dar valor a los atributos de un objeto Chora considerando las excepciones especificadas. **(0,75 puntos)**
  - El **operador "<<"** sobrecargado para mostrar por pantalla el contenido de un objeto Chora. **(0,5 puntos)**

UNIVERSIDAD NEBRIJA

Ejercicio Nº2 (4 puntos)

Crea una clase "Vector" que represente un vector matemático en 3 dimensiones. Esta clase debe tener como atributos privados tres valores enteros (x, y, z) y los siguientes métodos públicos:

- Un constructor que acepte tres valores enteros, uno para cada eje. **(0.25 puntos)**
- Sobrecarga el constructor para utilizar los valores por defecto (0, 0, 0) si n ose especifican los **(0.25 puntos)**
- Getters y setters para cada atributo. **(0.25 puntos)**
- Un método para calcular la magnitud $\|\vec{v}\|$ del vector, entendida como la parte positiva de la raíz de la suma de los cuadrados. **(0.5 puntos)**

$$\|\vec{v}\| = \sqrt{x^2 + y^2 + z^2}$$

- Sobrecarga del operador outstream (<<) para que se pueda hacer un cout de los vectores. **(0.75 puntos)**

Implementa un control de excepciones en los métodos setters, de manera que se lance una excepción si se intenta crear o establecer un valor con módulo superior a 999. **(1 punto)**

Escribe un programa principal en C++11, ***Ejercicio2,*** que cree varios vectores (al menos 5), calcule la magnitud de todos los vectores y maneje cualquier excepción que pueda ocurrir. Finalmente, intenta especificar el último vector fuera del rango válido. **(1 punto)**

UNIVERSIDAD NEBRIJA

Ejercicio Nº3 (3 puntos)

Escribe un programa en C++11, ***Ejercicio3,*** que cumpla las siguientes especificaciones.

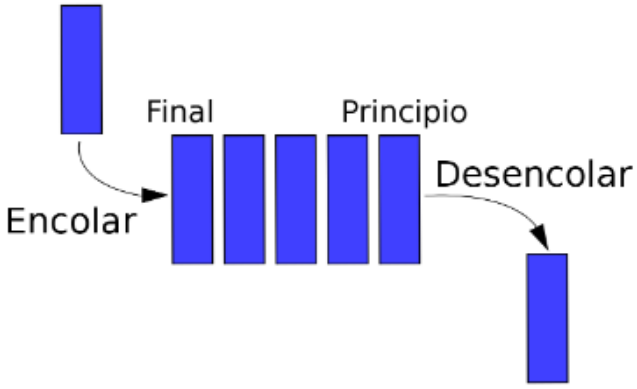
A partir del tipo de datos ***std::queue,*** cuya ayuda se adjunta al final del enunciado de este ejercicio, se pide implementar una clase templatizada de etiqueta ***MiVector*** que se comporte de forma similar al tipo de datos std::vector del estándar.

De manera que el tipo de datos que debéis implementar en este ejercicio, ***MiVector,*** debe incorporar:

- Constructor por defecto y constructor parametrizado con dos argumentos. El primero será el número de elementos que vamos a almacenar y el segundo será el valor del elemento en cuestión, con el que deberán ser inicializadas todas las posiciones del tipo contenedor std::queue que está encapsulado en el tipo de datos ***MiVector***. **(0.5 puntos)**
- Método ***at(int)*** que permita acceder al contenido del tipo de datos ***MiVector*** en la posición indicada mediante su argumento de tipo int. El método debe gestionar de manera adecuada los intentos de acceso a posiciones fuera de rango **(1 punto)**
- Método push_back(). **(0.25 puntos)**
- Método pop_back(). **(0. 25 puntos)**

Sobrecarga del operador << para mostrar el contenido almacenado de al menos una variable tipo ***MiVector<int>***, ***MiVector<string>*** y ***MiVector<perro>,*** siendo el tipo de datos perro definido por la siguiente estructura **(1 punto)**

```
struct perro{
        string nombre;
        int chip;
        int edad;
};
```



std::queue<T> es un tipo de datos contenedor que encapsula una cola.

Siendo algunos de sus métodos:

Functions

Nombre	Descripción
back	Devuelve una referencia al último elemento que se ha agregado más recientemente en la parte trasera de `queue`.
empty	Comprueba si la `queue` está vacía.
front	Devuelve una referencia al primer elemento en la parte delantera de `queue`.
pop	Quita un elemento de la parte delantera de `queue`.
push	Agrega un elemento a la parte trasera de `queue`.