PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
2do. Examen
(Primer Semestre 2021)

Indicaciones Generales:

- Duración: 3 horas.

Obligatoriamente los alumnos deberán mantener en todo momento el audio de sus computadores abierto de modo que puedan recibir los comunicados que se hagan durante el examen. De tener algún problema deberán hacérselo saber de inmediato al profesor de su horario por correo. De no hacerlo, no se aceptarán reclamos alegando que no oyeron las indicaciones.

- No se pueden emplear **variables globales, estructuras** ni la clase String. Tampoco se podrán emplear las funciones malloc, realloc, strdup o strtok, igualmente no se puede emplear cualquier función contenida en las bibliotecas stdio.h, cstdio o similares y que puedan estar también definidas en otras bibliotecas. **SOLO SE PODRÁ HACER USO DE PLANTILLAS Y STL EN AQUELLAS PREGUNTAS QUE ASÍ LO SOLICITEN, DE LO CONTRARIO SE ANULARÁ LA PREGUNTA.**
- Deberá modular correctamente el proyecto en archivos independientes. LAS SOLUCIONES DEBERÁN DESARROLLARSE BAJO UN ESTRICTO DISEÑO DESCENDENTE. Cada función **NO** debe sobrepasar las 20 líneas de código aproximadamente. El archivo **main.cpp** solo podrá contener la función. En cada archivo que implemente en los proyectos (.h y .cpp) deberá colocar un comentario en el que coloque claramente su nombre y código, de no hacerlo se le **descontará 0.5 puntos por archivo (NO SE HARÁN EXCEPCIONES)**.
- El código comentado NO SE CALIFICARÁ.
- La cláusula **friend** solo se podrá emplear en el caso de clases autoreferenciadas para ligar el nodo con la clase **inmediata** que encapsula la lista, **en ningún caso adicional**. No se considerará en la nota las clases que violen esto. Tampoco se podrá emplear la cláusula **protected**.
- **Deberá mantener en todo momento el encapsulamiento de todos los atributos de las clases así como guardar los estándares en la definición y uso de todas las clases desarrolladas.**
- Salvo en la sobrecarga de los operadores >> y <<, **no se podrán definir funciones (ni plantillas de funciones) independientes que no estén ligadas como métodos a alguna de las clases planteadas.**
- **Todos los métodos y funciones ligados a una clase deben estar desarrollados en el mismo archivo.**
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no den resultados coherentes en base al 60%.
- Se tomará en cuenta en la calificación el uso de comentarios relevantes.

SE LES RECUERDA QUE, DE ACUERDO AL REGLAMENTO DISCIPLINARIO DE NUESTRA INSTITUCIÓN, CONSTITUYE UNA FALTA GRAVE COPIAR DEL TRABAJO REALIZADO POR OTRA PERSONA O COMETER PLAGIO. ESTO Y EL HECHO DE ENCONTRAR CUALQUIER ARCHIVO YA SEA .cpp O .h CON FECHA U HORA DE CREACIÓN ANTERIOR A LA EVALUACIÓN SERÁ CONSIDERADO UNA FALTA DE PROBIDAD Y POR LO TANTO AMERITARÁ LA ANULACIÓN DE LA PRUEBA.

NO SE HARÁN EXCEPCIONES ANTE CUALQUIER TRASGRESIÓN DE LAS INDICACIONES DADAS EN LA PRUEBA

Puntaje total: 20 puntos

Cuestionario:

Cree una carpeta denominada **"EXAMEN02-2020-2_INF281"**, dentro de ella colocará los proyectos que a continuación se solicitan. **DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ 3 PUNTOS DE LA NOTA FINAL**.

PARTE 1 (8 puntos) STL y Polimorfismo

Por disposición del Ministerio de Transportes se ha modificado el reglamento para la renovación de las licencias de los conductores, separándolos en 2 grupos por la numeración de sus licencias, todas aquellas que empiecen con el número 5 hacia adelante son consideradas **nuevas**, el resto son consideradas

antiguas (todas las licencias tienen 8 dígitos). Por ejemplo, el número 35912665 pertenece a una licencia antigua y el número 79766395 pertenece a una licencia nueva. Para renovar la licencia cada grupo tiene un conjunto de requisitos y pagos que realizar, a continuación, se pasan a detallar:

- **Para licencias antiguas:**
  - Pago por trámite de licencia S/. 300.00
  - Pago por examen médico S/. 100.00
  - Pago por examen de manejo S/. 500.00
  - El conductor debe pagar todas las multas que tiene registradas.
  - Si el conductor tiene 5 o más infracciones **Muy graves** no se le renovará la licencia.

- **Para licencias nuevas:**
  - Pago por trámite de licencia S/. 200.00
  - Pago por examen médico S/. 100.00
  - Pago por examen de reglas S/. 150.00
  - El conductor debe pagar solo las multas Muy Graves que tiene registradas.

Se solicita que desarrolle un proyecto denominado **"STL_Polimorfico"** dentro de la carpeta correspondiente, en el cual implemente una lista utilizando STL-list, donde almacene el pago que debe realizar cada conductor con el fin de renovar su licencia. Para esta tarea debe implementar las siguientes clases:

- **Para manejar los conductores**: La clase se denominará "**Conductor**" y deberá contener lo siguiente: 1) un atributo denominado **licencia**, este atributo debe almacenar la licencia del conductor (int), 2) un atributo denominado **nombre** definido por una cadena de caracteres (char*).
- **Para manejar las renovaciones**: La clase se denominará "**Renovacion**" y deberá contener lo siguiente: 1) un atributo denominado multas (double) que almacena el monto total a pagar por las infracciones cometidas, 2) un atributo denominado tramite (double) que almacenará el monto a pagar por el trámite de renovación de la licencia, 3) un atributo denominado exmedico (double) que almacenará el monto a pagar por el examen médico que debe pasar el conductor para renovar su licencia.
- **Para manejar las renovaciones antiguas**: La clase se llamará "**Antiguo**" y deberá contener lo siguiente: 1) un atributo denominado exmanejo (double) que almacenará el monto a pagar por el examen de manejo, que debe pasar el conductor para renovar su licencia, 2) un atributo denominado estado (int) que almacenará **0** si el conductor no puede renovar su licencia o un **número positivo** si es posible renovarla. Esta clase posee datos heredados de la clase "**Renovacion**".
- **Para manejar las renovaciones nuevas**: La clase se llamará "**Nuevo**" y deberá contener lo siguiente: 1) un atributo denominado exreglas (double) que almacenará el monto a pagar por el examen de reglas, que debe pasar el conductor para renovar su licencia. Esta clase posee datos heredados de la clase "**Renovacion**".
- **Para manejar los Nodos**: La clase se denominará "**Nodo**" y deberá contener lo siguiente: 1) un atributo denominado pcosto, este atributo es un puntero de la clase **Renovacion**, 2) un atributo denominado chofer perteneciente a la clase **Conductor**.
- **Para manejar la lista**: La clase se denominará "**Procesa**" y deberá contener lo siguiente: 1) un atributo denominado lrenueva, este atributo es un STL-list de la clase **Nodo**.

"**DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS**"

Con las clases indicas debe realizar las siguientes operaciones:

- En la clase **Procesa** debe implementar el método **lee**, que se encargará de la lectura del archivo "Conductores.csv" y carga la información de cada conductor en los nodos de la clase **Nodo**, utilizando la STL-list para generar una lista de conductores y sus respectivos pagos por renovación de licencia.
- De acuerdo con el tipo de licencia a ser renovada (si empieza con 5 a más es nuevo, en caso contrario es antiguo) debe cargarla clase correspondiente a la renovación, utilizando el puntero pcosto de la clase **Nodo**, el tipo de renovación **no** se colocará en la estructura, debido a que las renovaciones

deberán ser manipuladas con polimorfismo. Los datos correspondientes a las multas deben leerse del archivo "RegistroConsolidado.csv", el cual puede leerse varias veces.

```
RegistroConsolidado.csv
15029228,201,Leve,158
48450395,151,Grave,316
73357516,151,Grave,316
…
```

En cada línea aparece el número de la licencia, el código de la infracción, el tipo y el monto de la multa.

- Finalmente, en la clase **Procesa** debe implementar el método **imprime,** que se encargará en primer lugar de ordenar la STL-list de acuerdo con el número de licencia del conductor y en segundo lugar debe realizar la impresión de un archivo con el monto a pagar por cada conductor de acuerdo con su tipo de renovación (**sin usar el carácter '\t'**), como se muestra en la parte inferior:

```
                                          Renovación de Licencias
===============================================================================================================

Licencia    Nombre                  Observación       Multas     Tramites     Ex.Med     Manejo/Reglas      Total
12270502    CUEVA FUENTES DELIA   No puede renovar
12443643    LEE SERRANO ROSARIO           -             6557        300          100               500    7457.00
12858682    ZARATE PEREZALEX              -             6241        300          100               500    7141.00
…           …
50200823    LEYVA SAM DELIA               -            11850        200          100               150   12300.00
50201756    ZEGARRA PENA CARMEN           -             5925        200          100               150    6375.00
51395324    TRIGUEROS ARTURO              -                0        200          100               150     450.00
…           …
```

Consideraciones:

Se le recomienda revisar al detalle los archivos csv, indicados. Para el desarrollo de la pregunta debe considerar el siguiente código:

```
#include "Procesa.h"
using namespace std;
int main(int argc, char** argv) {
    Procesa pro;

    pro.lee();
    pro.imprime();
    return 0;
}
```

No puede cambiar este código

PARTE 2 (12 **puntos**) **Plantillas de clases**

Emplear plantillas pre elaboradas como las STL es bueno, pero para un desarrollador, emplear siempre "cajas negras" nunca será lo ideal, conocer el funcionamiento interno de estos elementos puede llevar a descubrir deficiencias y a plantear mejoras. La intención de esta pregunta es tratar de que usted **emule** una STL, en particular un plantilla de tipo vector.

Parte A (5 puntos) (No puede emplear aquí punteros void ni punteros a funciones)

Cree un proyecto denominado **"Arreglo_Generico_EX01_PRO2A"**. El proyecto deberá definir una plantilla de clase denominada "Arreglo" que permita definir un arreglo dinámico que pueda contener cualquier tipo de dato, no puntero, desde valores enteros hasta objetos de cualquier tipo. El arreglo inicialmente no tendrá elementos y, según las necesidades, se irá incrementando sus espacios, este incremento será siempre duplicando su capacidad.

La plantilla estará compuesta por cuatro atributos: un puntero de tipo genérico (no void*) denominado **vector** que apuntará al espacio de memoria que contendrá a los datos, un valor entero **numElem**, que contendrá la cantidad de elementos que tenga el arreglo, un valor entero, **cap**, que contendrá la capacidad de contenedor. Debe respetar estos nombres.

La plantilla deberá contener un constructor, un destructor, métodos selectores que se requieran además de los siguientes métodos:

operator =, copia un arreglo en otro (arreglo1 = arreglo2)

push_back (dato), debe poder agregar el dato al final de los datos del arreglo, si el arreglo no puede contenerlo, deberá duplicar su capacidad.

push_front (dato), debe poder agregar el dato al inicio de los datos del arreglo, desplazando el resto, si el arreglo no puede contenerlo, deberá duplicar su capacidad.

at(i), la función solo debe devolver el elemento i del arreglo, **pero como una referencia T &** (igual como se hace en la sobrecarga de << o >>, al devolver la variable de archivo). Esto permitirá asignar un valor al elemento i del arreglo o usarlo para obtener el valor del dato i del arreglo.
Por ejemplo:
arr.at(3) = V; permitirá asignar **V** al elemento *i* del arreglo, donde **V** es una variable o valor constante de tipo T.
V = arr.at(3); permitirá asignar a **V** el elemento **i** del arreglo, donde **V** es una variable de tipo T.
Si el índice i está fuera de los límites del arreglo debe enviar el mensaje "ERROR DE ASIGNACION" y el programa debe truncarse.

resize(n), mueve los datos a un espacio de n elementos, si n es menor que la capacidad solo se queda con los primeros n elementos.

shrink_to_fit(), mueve los datos contenidos en el arreglo a un espacio exacto de memoria.

sort(ini, fin) perite ordenar el arreglo desde el índice ini hasta el índice fin. Debe emplear el método QuickSort o similar de eficiencia O(n log n).

Deberá probar todos los elementos de la plantilla para un arreglo que contenga valores de punto flotante. De no colocar una especificación para la plantilla, se considerará, al momento de la calificación, que el proyecto NO COMPILA y se calificará como tal.

Parte B (7 puntos)
Se tienen dos archivos del tipo CSV, los cuales se describen a continuación:

```
ConductotresP2.csv
30018447,FUENTES QUISPE CINTHIA DELIA
32517791,ZORRILLA LARA ARTURO
…
```

licencia, nombre del conductor

```
RegistroDeFaltas.csv
30018447,O1C-880,25/8/2020,158.50
78153392,A7R-205,18/12/2017,316.75
…
```

licencia, placa, fecha, monto de infracción

Defina las siguientes clases:

Falta
char *placa; int fecha; int multa;

Conductor
int licencia; char*nombre; Arreglo<Falta> falta;

RegistroDeFaltas
Arreglo <Conductor> conductores;

NO puede modificar, borrar o agregar atributos.

La implementación contemplará la definición de atributos, constructores por defecto, constructores copia, destructores, métodos selectores, sobrecarga del operador de asignación y todos métodos y sobrecargas de operadores que requiera (se sugiere también sobrecargar el operador de asignación: conductor = falta, que agregue una falta al final del Arreglo falta). Debe emplear obligatoriamente los nombres indicados y el método **at**.

La clase RegistroDeFaltas deberá desarrollar las siguientes tareas:
- Leer los datos del archivo ConductoresP2.csv y llenar el arreglo faltas.
- Leer los datos de RegistroDeFaltas.csv y complete los datos de las faltas de cada conductor.
- Mostrar un reporte como el que se indica a continuación:

```
                         Relación de Infractores
=====================================================================
No.        Licencia       Conductor                    Total a pagar
---------------------------------------------------------------------
  1)       48308899       Chung Tanaka Zkenia Irma        12456.97
  2)       66091742       Prado Quispe Carlos Julian       9365.50
  …        …              …                            …
=====================================================================
CONDUCTOR MAYOR MONTO A PAGAR POR MULTAS:
53682355   Valdes Quispe Rodolfo Pedro                    35766.37
Faltas del conductor:
No.        Placa          Fecha                           Monto
---------------------------------------------------------------------
  1)       M2F1239        20/5/2021                       456.97
  2)       K0W333         5/3/2020                        399.50
  …        …              …                               …
```

Las faltas deben aparecer ordenadas descendentemente por el monto de la multa. Debe emplear obligatoriamente em método sort de la plantilla.

No se puede emplear el carácter '\t' en el reporte.

Al finalizar el examen, comprima la carpeta de su proyecto empleando el programa Zip que viene por defecto en el Windows, **no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares**. Luego súbalo a la tarea programa en Paideia para este examen.

Profesor del curso: Rony Cueva
Miguel Guanira

San Miguel, 20 de julio del 2021.