PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
2do.Examen
(Primer Semestre 2023)

Indicaciones Generales:

- Duración: 3 horas.

  SOLO ESTÁ PERMITIDO EL USO DE APUNTES DE CLASE. NO PUEDE UTILIZAR FOTOCOPIAS NI MATERIAL IMPRESO, TAMPOCO PODRÁ EMPLEAR HOJAS SUELTAS.

- No se pueden emplear **variables globales, estructuras, ni la clase (o el tipo de datos) 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 **(una línea de código no es una línea de texto, es una instrucción que termina con un punto y coma)**. El archivo **main.cpp** solo podrá contener la función **main** de cada proyecto y el **código contenido en él solo podrá estar conformado por tareas implementadas como métodos de clases**. En el archivo main.cpp deberá colocar un comentario en el que coloque claramente su nombre y código, **de no hacerlo se le descontará 0.5 puntos en la nota final. (NO SE HARÁN EXCEPCIONES)**.
- El código comentado NO SE CALIFICARÁ **y esto incluye el comentar el llamado a la función que lo contiene**.
- La cláusula **friend** solo se podrá emplear en el caso de clases auto referenciadas 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.**
- **NO PUEDE EMPLEAR ARREGLOS NI ARCHIVOS AUXILIARES PARA COLOCAR PARTE O TODOS LOS DATOS DE LOS ARCHIVOS DADOS**
- 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.
- **NO PUEDE UTILIZAR VARIABLES ESTÁTICAS.**

  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.

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

Puntaje total: 20 puntos

INDICACIONES INICIALES

- La unidad de trabajo será **t:** (Si lo coloca en otra unidad, no se calificará su laboratorio y se le asignará como nota cero)
- Cree allí una carpeta con el nombre **"CO_PA_PN_EX_FINAL_2023_1"** donde **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno y **PN** primer nombre (de no colocar este requerimiento se le descontará 3 puntos de la nota final). **Allí colocará los proyectos solicitados en la prueba.**

PREGUNTA 1 (12 puntos)

PARTE 1 (5.5 puntos)

Elabore un proyecto denominado "Pregunta01_Parte01_Alumnos_Notas" y en él desarrollará el programa que dé solución al problema planteado. DE NO COLOCAR ESTE REQUERIMIENTO SE LE DESCONTARÁ DOS (2) PUNTOS DE LA NOTA FINAL.

Una institución de educativa desea una aplicación orientada a objetos que permita manejar las notas de sus alumnos. Para esto se tienen dos archivos del tipo CSV, los cuales se describen a continuación:

Alumnos-Preg01.csv

```
20082060,Erasmo Gomez Montoya
20082062,Marcia Guzman Moncada
……
```

Código del Alumno, Nombre del Alumno

Notas-Preg01.csv

```
20221, 20082060,INF281,5,18,IND131,3.5,8
20221, 20082062,INF281,5,20,IND132,4,20
20231, 20082060,IND131,3.5,11
…
```

Semestre dictado, código del alumno y lista de notas.
La lista de notas se identifica por: el código del curso, sus créditos y la calificación

Se le está proporcionando la biblioteca estática "**libbibliotecaarchivos.a**" y su respectivo .h con la implementación de las funciones de apertura de archivos, deberá incorporar OBLIGATORIAMENTE estos archivos al proyecto y utilizarlos para solucionar el problema planteado a continuación, DE LO CONTRARIO SE DESCONTARÁ 1 PUNTO POR CADA APERTURA DE ARCHIVOS QUE NO PROVENGAN DE LA LIBRERÍA.

El proyecto deberá definir tres clases como se muestran a continuación:

Nota
int **ciclo;** char* **codigo_curso**; double **creditos**; int **calificacion;** int **vez;**

Alumno
int **codigo_alumno**; char* **nombre**; double **promedio**;

AlumnoNotas
Alumno alumno; vector<class Nota>**notas**;

Facultad
ListaDoble alumnosxNotas;

Nodo
AlumnoNotas dato; Nodo *siguiente; Nodo *anterior;

ListaDoble
Nodo *inicio Nodo *fin

NO PUEDE MODIFICAR, BORRAR O AGREGAR ATRIBUTOS

En la **clase Alumno** debe definir todos los métodos que sean necesarios, pero será obligatorio que implemente el operador **>>** que lea un alumno desde Alumnos-Preg01.csv (codigo_alumno y nombre) y los asigne a los atributos correspondientes. Además, debe implementar un método **void imprimeAlumno (ofstream &)** el método recibe la variable de archivo y muestre ambos atributos en una sola línea y bien tabulados.

En la **clase Nota** debe definir todos los métodos que sean necesarios, pero será obligatorio que implemente el operador **>>** que solo lea tres datos del archivo Notas-Preg01.csv (codigo_curso, créditos y calificación) y los asigne a los atributos correspondientes ((el ciclo se leerá y asignará de manera externa a este operador). Además, debe implementar un método **void imprimeNota (ofstream &)** el método recibe la variable de archivo y muestre todos los atributos en una sola línea y bien tabulados.

Para la **clase Facultad** deberá implementar como mínimo los siguientes métodos:

void leerAlumnosNotas (const char*, const char *); El método lee los datos del archivo "Alumnos-Preg01.csv" y los inserte en la lista doblemente ligada, ordenado por **el código del alumno**, luego debe leer los datos del archivo "Notas-Preg01.csv" y llena el contenedor "vector<class Nota>notas". Recibe como argumento los nombres de los archivos. En esta operación deberá verificar que **NO SE AGREGUEN ALUMNOS REPETIDOS AL CONTENEDOR. Debe incluir la nota al final del vector de notas.** LOS ARCHIVO SOLO SE PUEDEN LEER UNA VEZ.

void imprimirAlumnosNotas (const char*); El método imprime TODOS los datos almacenados la lista doblemente ligada. Recibe como argumento el nombre del archivo del reporte. Debe colocar un título general y encabezados encima en cada columna. Debe alinear y formatear correctamente los encabezados.

Para manejar los Nodos: La clase se denominará **"Nodo"** y deberá contener lo siguiente: 1) un atributo denominado dato de la clase AlumnoNotas, 2) un atributo denominado siguiente, este atributo es un puntero a la clase Nodo (autoreferenciado), 3) un atributo denominado anterior, este atributo es un puntero a la clase Nodo (autoreferenciado).

Para manejar la lista doble: La clase se denominará "**ListaDoble**" y deberá contener lo siguiente: 1) un atributo denominado inicio, este atributo es un puntero de clase Nodo, 2) un atributo denominado fin, este atributo es un puntero de clase Nodo. La lista doble deberá estar ordenada por código del alumno y ciclo (ascendente). Esta estructura es una lista doblemente enlazada.

Para manejar la facultad: La clase se denominará "**Facultad**" y deberá contener lo siguiente: 1) un atributo denominado **alumnoxnotas**, este atributo de la clase **ListaDoble**, donde se guardarán toda la información de los alumnos y sus notas.

Además, usted podrá definir todos los métodos que crea conveniente para las clases Facultad, **AlumnoNotas**, **Alumnos** y **Notas** de modo que pueda solucionar el problema.

El código de la función main será el siguiente:

```
#include "Facultad.h"
int main(int argc, char** argv) {
   class Facultad  facultad;
   facultad.leerAlumnosNotas"Alumnos Preg01.csv","Notas Preg01.csv");
   facultad.imprimeAlumnosNotas("PruebaAlumnosNotas.txt");

   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO, DE HACERLO RECIBIRÁ UN DESCUENTO DE 2 PUNTOS EN LA NOTA FINAL DE LA PREGUNTA

- LOS ARCHIVOS CSV SOLO SE PUEDEN LEER UNA VEZ.
- EN LOS REPORTES NO PODRÁ EMPLEAR EL CARÁCTER '\t'.
- NO PUEDE COLOCAR LOS DATOS DEL ARCHIVO EN ARREGLOS o ESTRUCTURAS AUTOREFERENCIADAS AJENAS A LOS CONTENEDORES DADOS.

PUNTAJES PARTE 1:

a. Lectura, asignación y pruebas de Alumno: 2.0 puntos.
b. Lectura y asignación correcta de la lista de notas: 2.5 puntos.
c. Reporte de los alumnos y sus notas: 1.0 puntos.

NO SE ASIGNARÁ PUNTAJE A LOS REPORTES SI NO SE IMPLEMENTAN DE FORMA CORRECTA LOS PROCESOS DE CARGA O ACTUALIZACIÓN.

PARTE 2 (6.5 puntos)

Se solicita que desarrolle un proyecto denominado **"Pregunta01_Parte02_Actualiza_Notas"** dentro de la carpeta correspondiente. Para esto copie el proyecto de la Parte 1 con el nombre indicado. **NO SE CALIFICARÁ ESTA PARTE SI LA DESARROLLA EN EL PROYECTO DE LA PARTE 1**

Para la clase Facultad debe definir un método **actualizaNotas()** que permita actualizar y totalizar las notas de cada uno de los alumnos.

Para la clase **AlumnoNotas** debe también definir un método **actualizar()**; que permita eliminar los cursos repetidos y solo mantener en el vector las notas más recientes, es decir, si un alumno lleva por segunda vez en el ciclo 20231 el curso INF281 solo debe aparecer esa información en el vector, la información previa de ciclos anteriores para ese mismo curso NO debe aparecer en el vector resultante.

Para la clase **AlumnoNotas** debe definir el método **totalizar();** el método debe usar el **vector** notas y con él calcular el promedio ponderado de notas del alumno y llenar el atributo correspondiente.

El código de la función main será el siguiente:

```
#include "Facultad.h"
int main(int argc, char** argv) {
   class Facultad  facultad;
   facultad.leerAlumnosNotas“Alumnos Preg01.csv”,“Notas Preg01.csv”);
   facultad.imprimeAlumnosNotas("PruebaAlumnosNotas.txt");
   facultad.actualizaNotas();
   facultad.imprimeAlumnosNotas("PruebaNotasActualizadas.txt");
   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO, DE HACERLO RECIBIRÁ UN DESCUENTO DE 2 PUNTOS EN LA NOTA FINAL DE LA PREGUNTA

PUNTAJES PARTE 2: (6.5 puntos)

a. Implementar el método actualizar: 5.0 puntos.
b. Implementar el método totalizar: 1.5 puntos.

PARA LA ASIGNACIÓN DEL PUNTAJE DE LOS PROCESOS DE CARGA O ACTUALIZACIÓN DEBE MOSTRARSE LA INFORMACIÓN ACTUALIZADA EN LOS REPORTES CORRESPONDIENTES.

PREGUNTA 2 (8 puntos) Autoreferencia y Polimorfismo

Se solicita que desarrolle un proyecto denominado **“STL_Polimorfismo_Pregunta02”** dentro de la carpeta correspondiente, con el fin de registrar los créditos que ha llevado un alumno. Para esta tarea debe implementar las siguientes clases:

- **Para manejar los cursos por alumno:** La clase se denominará **“CursoAlumno”** y deberá contener lo siguiente: 1) un atributo denominado **codigo** (**int**), 2) un atributo denominado **codcur** definido por una cadena de caracteres dinámica que representa el código del curso, 3) un atributo denominado **ciclo** (**int**), 4) un campo denominado **creditos** (**double**) que representa la cantidad de créditos que tiene un curso, 5) un campo denominado **nota** (**int**), 6) un campo denominado **vez** (**int**) la vez que está llevando el curso el alumno.
- **Para manejar los cursos en primera:** La clase se denominará **“Primera”** y deberá contener lo siguiente: 1) un atributo denominado **codacceso** definido por una cadena de caracteres dinámica que representa un código para ingresar al curso de forma virtual, el cual estará dado por el código del alumno concatenado con el código del curso. Esta clase posee datos heredados de la clase **CursoAlumno.**
- **Para manejar los cursos en segunda:** La clase se denominará **“Segunda”** y deberá contener lo siguiente: 1) un campo denominado **creditos** (**double**) que representa la cantidad de créditos extra que debe pagar un alumno por llevar un curso en segunda. Esta clase posee datos heredados de la clase **CursoAlumno.**
- **Para manejar los cursos en tercera:** La clase se denominará **“Tercera”** y deberá contener lo siguiente: 1) un campo denominado **porcentaje** (**double**) que representa el porcentaje de créditos extra que debe pagar un alumno por llevar un curso en tercera. Esta clase posee datos heredados de la clase **CursoAlumno.**
- **Para manejar las notas:** La clase se denominará **“Nota”** y deberá contener lo siguiente: 1) un atributo denominado **pnota,** este es un puntero de la clase **CursoAlumno.**
- **Para manejar el registro de notas**: La clase se denominará **“Registro”** y deberá contener lo siguiente: 1) un atributo denominado **vregistro**, este atributo es un STL-Vector de la clase **Nota**, donde se guardarán todas las notas que existen en el archivo a cargar, inicialmente todas las notas se cargaran como un curso que se lleva por primera vez.

“DEBE EMPLEAR OBLIGATORIAMENTE LOS NOMBRES DE LAS CLASES Y SUS ATRIBUTOS”

Con las clases indicas debe realizar las siguientes operaciones:

- (**1.0 punto**) En la clase **Registro** debe implementar el método **carga**, que se encargará de la lectura del archivo " registronotas.csv" y cargar la información en el vector **vregistro**. Las notas del alumno deben cargarse como un curso en **Primera** sin excepciones. Es opcional usar un método polimórfico.
- (**6.0 puntos**) En la clase **Registro** debe implementar el método **procesa** que debe realizar los siguientes pasos para una evaluación de los cursos llevados por un alumno de forma óptima:

- Debe ordenar el vector **vregistro** por código de alumno, código del curso y ciclo. Con este ordenamiento los cursos que ha llevado un alumno quedan juntos de tal forma que no es necesario recorrer todo el vector para evaluar si un curso ha sido llevado en primera, segunda o tercera.
- Una vez ordenado el vector, debe evaluar cada curso llevado por el alumno y verificar si lo ha aprobado en primera, segunda y tercera. Como esta ordenado **no debe recorrer todo el vector** para esta verificación.
- Si el curso lo aprobó en primera no debe realizar ninguna operación.
- Si verifica y el curso lo aprobó en segunda **debe dejar una sola posición** dentro del vector que represente al curso en segunda, desde luego debe realizar las operaciones necesarias ya que inicialmente todos los cursos se insertaron aprobados en primera. Como parte del proceso debe llenar el atributo **creditos** de la clase **Segunda**, con un crédito si el curso tiene un valor de 3 créditos o menos o 1.5 créditos en caso contrario. Recuerde actualizar el atributo **vez**.
- Si el curso lo aprobó en tercera **debe dejar una sola posición** dentro del vector que represente al curso en tercera, desde luego debe realizar las operaciones necesarias ya que inicialmente todos los cursos se insertaron aprobados en primera. Como parte del proceso debe llenar el atributo **porcentaje** de la clase **Tercera**, con 50% si el curso tiene un valor de 3 créditos o menos o 100% en caso contrario. Recuerde actualizar el atributo **vez**.

Para actualizar los atributos cuando el curso es aprobado en **Segunda** o **Tercera**, debe desarrollar un método polimórfico **actualiza**. Este método puede recibir los parámetros que crea necesarios.

- (**1.0 puntos**) Finalmente, en la clase **Registro** debe implementar el método **muestra**, que se encargará de realizar la impresión de un archivo con la información de los alumnos y sus cursos. Los créditos que se muestran en el reporte es el resultado de la operación del atributo base con el atributo determinado en la derivada (**para esta operación debe usar el método polimórfico imprime de la clase CursoAlumno**).

```
                                REGISTRO DE NOTAS
=============================================================================================================
 Codigo       Curso     Ciclo       Nota        Vez    Creditos   Codigo de Acceso
=============================================================================================================
 202119707    INF246    202302        17          3      8.00
 202123402    INF246    202202        16          2      5.50
 202123703    INF263    202302         9          2      6.00
 202123703    MAT241    202202        13          1      4.00     202123703MAT241
 202123703    MEC270    202302        14          1      3.50     202123703MEC270
 202125607    MEC206    202201        19          1      3.00     202125607MEC206
 202125775    FIS220    202202        19          2      5.00
 202125775    MEC206    202201        18          1      3.00     202125775MEC206
...
```

Consideraciones:

- El main debe contar con la siguiente información:

```
#include "Registro.h"
using namespace std;

int main(int argc, char** argv) {
   Registro reg;

   reg.carga();
   reg.procesa();
   reg.muestra();

   return 0;
}
```

NO PUEDE CAMBIAR ESTE CÓDIGO

- Para convertir un número a cadena se le recomienda emplear la función **sprintf** de la biblioteca **cstring**. Esta función tiene el siguiente formato: **sprintf**(cadena,"%d",numero).
- Se le recomienda revisar al detalle los archivos csv, indicados a continuación:

registronotas.csv

```
202216453,MEC206,3,202102,8
202119707,INF246,4,202302,17
```

```
202119707,MEC289,3,202202,7
…
Cod Alumno,Cod Curso,Créditos,Ciclo,Nota
```

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
Erasmo Gómez
Miguel Guanira

San Miguel, 11 de julio del 2023.