PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1

2do. Examen
(Primer Semestre 2020)

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.
- De no respetarse el nombre de los proyectos, carpetas, variables o funciones indicadas en la prueba se les descontará puntaje por cada trasgresión a criterio del profesor. Esta sanción podría llegar a la anulación del examen si se sospecha una falta de probidad.
- Cree en el computador una carpeta de trabajo con el siguiente nombre: ***Examen02-2020-01***. En ella colocará los proyectos que den solución a los problemas planteados.
- Deberá modular correctamente el proyecto en archivos independientes. LAS SOLUCIONES DEBERÁN DESARROLLARSE BAJO UN ESTRICTO DISEÑO DESCENDENTE. Cada módulo **NO** debe sobrepasar las **20 líneas de código** aproximadamente. El archivo **main.cpp** solo podrá contener la función **main** de cada proyecto y el código contenido en ella solo podrá estar conformado por mensajes a objetos.
- **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.**
- 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 clase que violen esto
- **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.**
- El código comentado NO SE CALIFICARÁ.
- 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 AL EXAMEN 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:

PARTE 1 (**7 puntos**) **Herencia y Polimorfismo**

Se desea elaborar una aplicación orientada a objetos que permita procesar la información de artículos académicos (publicaciones o *papers* en inglés), para poder generar las citas bibliográficas requeridas al momento de escribir un trabajo académico que deba referenciarlos. Estos artículos pueden provenir de diversas fuentes en donde son publicadas, y entre ellas, se encuentran dos tipos muy comunes:

Actas de Conferencias: Los artículos son presentados en eventos/conferencias que se realizan en diversas ciudades/países cada cierto tiempo. Al final, estos artículos son publicados en sus actas (*proceedings*). Un ejemplo de una posible referencia a estos artículos se muestra a continuación:

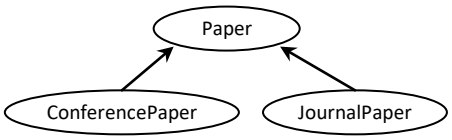
> Oncevay, Arturo (2017). Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language. International Conference on Text, Speech, and Dialogue. Czech Republic.
>
> **<Author_LastName, Author_FirstName> (<Publication_Year>). <Paper_Title>. <Conference_Name>. <Host_Country>.**

Revistas: Los artículos son enviados a publicación directa en revistas académicas. Estas revistas tienen diversos volúmenes (generalmente anuales). Un ejemplo de una posible referencia a estos artículos se muestra a continuación:

> Paz, Freddy (2019). Application of a New Questionnaire to Measure the Usability: A Case Study in the E-Commerce Domain. International Journal of Advanced Trends in Computer Science and Engineering, 8.
>
> **<Author_LastName, Author_FirstName> (<Publication_Year>). <Paper_Title>. <Journal_Name>, <Volume>**

En este sentido, se solicita al usuario implementar un determinado conjunto de clases, cuyas funcionalidades se irán extendiendo de manera progresiva, para completar la aplicación deseada.

Como se indicó, trabajaremos solo con 2 tipos de artículos (*Paper*), los cuales pueden ser de conferencias (*ConferencePaper*) o de revistas (*JournalPaper*). Como ambos tipos poseen datos comunes, pero también datos particulares a su clase, se requiere desarrollar una jerarquía de clases como se muestra en la figura.



Los datos miembros o atributos para estas clases serán los siguientes:

Paper
int id
char * title
char * author_name
int publication_year

ConferencePaper
char * conference_name
char * host_country

JournalPaper
char * journal_name
int volume

Paper: código numérico, título de la publicación, nombre del autor principal (en formato: "Last_name, First_name"), y el año de la publicación como entero.
ConferencePaper: nombre de la conferencia y el país o ciudad donde se organizó el evento.
JournalPaper: nombre de la revista y número del volumen.

Se pide que desarrolle un proyecto denominado **"Parte1-Herencia_Y_Polimorfismo"** en donde se declaren las tres clases descritas con las relaciones necesarias que permitan manipularlas empleando polimorfismo (no podrá modificar, agregar o eliminar los atributos).

Todas las clases deben tener implementado los siguientes métodos:

- ✓ Métodos Constructores por defecto (sin argumentos) y estándar (recibe todos los valores de los atributos como argumentos), y Destructor
- ✓ Métodos Get y Set para todos los atributos
- ✓ Un método "**leerDatos**" que permita leer los datos desde un archivo de texto (**ifstream**) que recibirá como argumento
- ✓ Un método "**citar**" que genere e imprima la referencia bibliográfica, según el formato indicado al inicio, en un archivo de texto (**ofstream**) que recibirá como argumento.

Asimismo, debe considerar lo siguiente:

- ✓ La memoria asignada a las cadenas de caracteres debe ser óptima.
- ✓ La clase Paper debe ser abstracta.
- ✓ Los métodos "**leerDatos**" y "**citar**" deben considerar el principio de **polimorfismo**.

La función principal (en el archivo "main.cpp) deberá probar de manera sencilla la implementación de estas clases y su manejo para el archivo de pruebas "**Papers.csv**", donde los datos están separados por "**;**" (punto y coma):

Papers.csv	
1	C;1;Oncevay, Arturo;2017;Ship-LemmaTagger: Building an NLP Toolkit for a Peruvian Native Language;International Conference on Text, Speech, and Dialogue;Czech Republic
2	J;2;Paz, Freddy;2019;Application of a New Questionnaire to Measure the Usability: A Case Study in the E-Commerce Domain;International Journal of Advanced Trends in Computer Science and Engineering;8
3	…..
El archivo contiene lo siguiente por línea: tipo del Paper (C: Conferencia, J: Journal), código (id) del Paper, nombre del autor (en formato "Apellido, Nombre"), año, título del Paper. Además, si es ConferencePaper (C): nombre de la conferencia, país. Por otro lado, si es JournalPaper (J): nombre de la revista, volumen.	

Aproveche los caracteres del inicio de línea para identificar qué tipo de objetos debe instanciarse y leerse, ya que en la prueba debe usar un arreglo de punteros a "**Paper**" para poder usar el polimorfismo adecuadamente: "**Paper * lstPapers[100]**" (no será necesario ajustar la memoria a tamaño exacto).

PARTE 2 (13 **puntos**) **Plantillas y STL**

Una entidad bancaria requiere una aplicación que le permita manejar los movimientos de entrada y salida efectuados en las diferentes cuentas de sus clientes. Para solucionar este problema deberá desarrollar los siguientes proyectos

a) (5 puntos) Elabore un proyecto denominado **"Parte2A-Plantilla"**. El proyecto deberá definir una plantilla de clases denominada ***ArbolBBG*** que permita definir un árbol binario de búsqueda genérico (ABBG). La plantilla deberá poder crear, insertar un nuevo nodo, imprimir de manera ordenada los datos almacenados y eliminar todo el árbol manualmente y por el destructor. Además, debe incluir un método denominado "**modificar**" que reciba únicamente dos argumentos: un valor ***entero*** y un valor de ***punto flotante***. El valor entero será el valor a buscar en el árbol, el valor de punto flotante servirá para modificar el dato del árbol, esto se realizará mediante el operador += (dato += valorPF). El método no devuelve resultados. La impresión de los datos deben estar ordenados y correctamente tabulados (no podrá emplear el carácter '\t').
   Para probar su plantilla puede utilizar cualquier dato que le simplifique esta operación. **De no colocar una especificación para la plantilla se considerará, al momento de la calificación, que el proyecto NO COMPILA**.

b) (8 puntos) Elabore un proyecto denominado **"Parte2B-Banco"**. Copie en él la plantilla desarrolladas en la parte "a" e incorpórela al proyecto, cualquier modificación o desarrollo que haga en este proyecto a la plantilla incorporada no modificará la calificación ni dará puntaje a la pregunta anterior, **NO SE HARÁN EXCEPCIONES**.
   La entidad bancaria cuenta con los siguientes archivos CSV:

Clientes.csv

```
71067314,Diaz Correa Irma Virginia Maria,2595.32
58673328,Chung Lee Sandro,16518.00
46116115,Cabrea Luna Shirley Isabel,23564.76
……
```

En cada línea aparecen los datos de cada cuenta, esto es el código de la cuenta, el nombre del cliente y el saldo inicial de la cuenta. Todas las cuentas están en moneda nacional.

TipoDeCambio.csv

```
S,1,SOLES
$,3.394,DOLARES
&,3.723,EUROS
Y,0.031,YEN
U,0.051,RUBLO
P,0.06,PESO
B,0.823,BOLIVIANO
…,…,…
```

Contiene los equivalentes en moneda nacional de cada moneda. En cada línea aparece el carácter de identificación de la moneda, su equivalente en moneda nacional y el nombre de la moneda.

Transacciones.csv

```
64046470,$3254.28,D,Y2133.64,R,$4514.11,D,S1440.48,D,S1006.9,D,&687.62,R
20652412,&1145.65,R
73012493,Y2747.71,R,Y12875.98,D,Y1256.88
……
```

En él archivo se almacenan las transacciones hechas por los clientes (depósitos y retiros) en un período de tiempo, en cada línea tenemos: el código de la cuenta seguido de una serie de operaciones. Cada operación está dada por el monto de la transacción, este valor estará precedido por un carácter, dependiendo del tipo de moneda en que se hizo la transacción (S = soles, $ = dólares americanos, & = euros, Y = yenes, etc.) y el tipo de transacción (D: depósito, R: retiro).

El proyecto deberá definir tres clases, los tipos de datos asociados a los atributos se indican a continuación (no podrá modificarlos ni agregar más):

Clase Cuenta
int codigo char *cliente; double saldo; char *estado; (*)

Clase Moneda
char simbolo; double equivalencia; char *nombre;

La implementación contemplará la definición de atributos, constructores por defecto, constructores copia, destructores, métodos selectores, un método que permita leer los datos desde un archivo de texto, otro que permita imprimirlo en otro archivo de texto. Además las sobrecargas necesarias para el manejo de plantillas. La memoria asignada a las cadenas debe ser exacta.

(*) Inicialmente el atributo "estado" contendrá la palabra "HABIL".

Clase Cuentas
class ArbolBBG<class Cuenta> arbClientes; map<char,class Moneda> monedas;

La clase deberá contar con los siguientes métodos:

leerCuentas: recibirá como parámetro una cadena de caracteres con el nombre del archivo de clientes, el método deberá llenar el árbol con todos los clientes del banco.

leerTiposDeCambio: deberá cargar el contenedor con los datos del archivo con los tipos de cambio.

ejecutarMovimientos: deberá actualizar las cuentas del banco con las operaciones registradas en el archivo de transacciones. Si el saldo de la cuenta se torna negativo el estado debe cambiarse a "SOBREGIRADO", si se torna a positivo debe volver a "HÁBIL".

reporteDeCuentas: recibirá como parámetros una cadena de caracteres con el nombre del archivo de reporte y un carácter ('C' o 'A'), el método deberá poder crear ('C') el archivo o agregar ('A') a un archivo un reporte según el formato indicado adelante.

La función main del proyecto deberá únicamente definir un objeto de clase Cuentas y empleado solo sus métodos imprimir un reporte como el que se indica a continuación:

```
                              ESTADOS DE CUENTAS
================================================================================
                          SALDO INICIAL DE LAS CUENTAS
================================================================================
Codigo     Cliente                                 Saldo inicial   Estado
--------------------------------------------------------------------------------
11546673   Honores Garcia Pedro Juvenal               8923.80      HABIL
23564810   Patiño Quispe Maria Lucia Alejandra       23456.12      HABIL
44225511   Rosas Lopez Pedro Javier                    800.00      HABIL
…          …                                           …           …
================================================================================
                           SALDO FINAL DE LAC CUENTAS
================================================================================
Codigo     Cliente                                  Saldo final   Estado
--------------------------------------------------------------------------------
11546673   Honores Garcia Pedro Juvenal               -456.78     SOBREGIRADO
23564810   Patiño Quispe Maria Lucia Alejandra       23456.12     HABIL
44225511   Rosas Lopez Pedro Javier                    123.00     HABIL
…          …                                           …           …
================================================================================
```

Anotaciones finales

Al finalizar el laboratorio, comprima[1] la carpeta **Examen02-2020-01** y súbalo a Paideia.

El acceso para subir su examen a Paideia quedará cerrado automáticamente a las 11:00 a. m. por lo que deberán tomar las precauciones del caso para no sobrepasar este tiempo.

Profesor del curso: Rony Cueva
Miguel Guanira

San Miguel, 21 de julio del 2020.

[1] Para evitar problemas en la corrección de la prueba, utilice el programa de compresión que viene por defecto en el Windows (Zip) no 7z, no RAR, etc.