PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
2do. Examen
(Segundo 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. **SOLO SE PODRÁ HACER USO DE PLANTILLAS Y STL EN AQUELLAS PREGUNTAS QUE ASÍ LO SOLICITEN.**
- 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**.
- 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 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 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 "LP1_EXAMEN_02_2020_1", dentro de ella colocará los proyectos que a continuación se solicitan. **DEBE RESPETAR ESTE REQUERIMIENTO**.

PARTE 1 (8 puntos) Herencia y Polimorfismo

Se pide que desarrolle un proyecto denominado "Polimorfismo_Parte01" en donde se declaren las clases descritas con las relaciones necesaria que permitan manipularlas empleando polimorfismo.

En la institución de salud pueden ser atendidos tres tipos de pacientes: ambulatorios, de urgencias y de emergencias. Estos tipos poseen datos comunes heredados de la clase Paciente, y también datos particulares en su propia clase. Por tal motivo se requiere desarrollar las siguientes clases como se muestra a continuación:

Paciente	Ambulatorio	Urgencia	Emergencia
int dni; char * nombre; char * codMed;	double porcSeguro;	double porcSeguro; double porcUrgencia;	int telefonoRef; char *nombreRef;

La implementación contemplará la definición de atributos, constructores por defecto, constructores copia, destructores, métodos selectores, luego los siguientes métodos polimórficos: un método que permita **leer** los datos desde un archivo de texto, otro que permita **imprimir** en otro archivo de texto, otro que devuelva la **prioridad** de atención de acuerdo con el tipo de paciente y la edad del paciente, se debe considerar como primer valor de ordenamiento el tipo de paciente (primero los de emergencia, luego los de urgencia y finalmente los ambulatorios) y como segundo valor de ordenamiento la edad, en orden descendente (primero las personas de mayor edad), finalmente otro método que devuelva una **observación**, si un paciente es de riesgo (mayores de 65 años) y para el caso de los pacientes de urgencia o ambulatorio, si es menor de edad ya que deben venir con un apoderado como referencia. La memoria asignada a las cadenas debe ser óptima.

Luego se debe implementar la clase **NPaciente** que tiene como único atributo un puntero a la clase **Paciente**. Además, debe implementar la clase **Procesa** que tiene como único atributo un arreglo fijo de la clase **NPaciente**, con un método **lee** que recibe como parámetro el archivo "AtencionMedicaP.csv" que contiene a todos los pacientes atendidos de acuerdo con las citas realizadas, el archivo es similar al siguiente:

Parte01_AtencionMedicaP.csv

```
A,12484697,Angel Shirakawa Ortega,70,OH6720V,37.94,A,13633991,Alexander Portugal Raffo,35,…
A,20371464,Felicita Villegas Leiva,19,PQ5152C,83.34,U,20688048,Humberto Alfonso …
…
```

También un método **ordena**, que se encarga de ordenar el arreglo de acuerdo con la prioridad del paciente para la atención dada por el método polimórfico **prioridad**. Finalmente, un método **imprime** que muestra un reporte de atención de los pacientes de acuerdo con el siguiente formato;

```
                                        INSTITUTO DE SALUD SA
=============================================================================================================
                                    Orden de atención de los pacientes
-------------------------------------------------------------------------------------------------------------

No.     DNI         Paciente                  Edad          Observacion                Referencia
  1)    48308899    Zkenia Irma Chung          94     Persona de Riesgo          Ana Perez      9112253
  2)    66091742    Carlos Julian Prado        93     Persona de Riesgo          Jose Quispe    7812354
  …     …           …                          …                                 …              …
 37)    57112777    Emilio Aguirre             90     Persona de Riesgo
  …     …           …
 69)    41964357    Virginia Salamanca         21
 70)    54815159    Annie Ofelia Asmad         20
 71)    39335774    Victoria Pomajambo         15     Debe venir con su apoderado
  …     …           …                          …
 75)    28992638    Eloy Jaime Ortega          94     Persona de Riesgo

=============================================================================================================

Pacientes: 121  Personas de Riesgo:39
Menores de Edad sin apoderado: 13
```

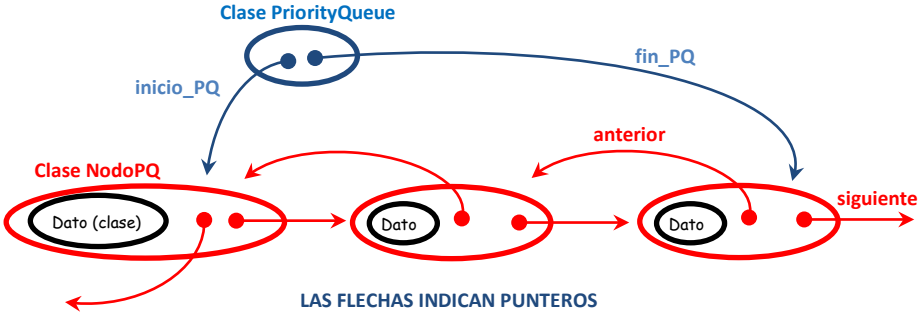
PARTE 2 (12 **puntos**) **Plantillas y STL**

Una cola de prioridad (**priority queue**) es una estructura de datos en la que los elementos se atienden en el orden indicado por una prioridad asociada a cada elemento. Si varios elementos tienen la misma prioridad, se atenderán de modo convencional dependiendo del momento de llegada a la cola. Por ejemplo si llegaran los siguientes pares de números a la cola: (2,3), (4,1), (2,1), (3,3), (1,1), (3,1), (2,4), (1,8) y (3,5), y la prioridad estuviera dada por el primer elemento del par, de modo que a mayor número mayor prioridad, los pares serían atendidos en el siguiente orden: (4,1), (3,3), (3,1), (3,5), (2,3), (2,1), (2,4), (1,1), (1,8). Para poder trabajar con una cola de prioridad (**PQ**) deberá desarrollar lo siguiente:

a) (5 puntos) Elabore un proyecto denominado **"Priotity_Queue_Parte02A"**. El proyecto deberá definir las plantillas de clases denominadas **PriorityQueue** y **NodoPQ** que permitan definir una cola de prioridad genérica (PQG). La plantilla deberá poder crear una **PQG**, realizar operaciones de llegada

(encolar), atención (desencolar) de datos y verificación si la cola está o no vacía. También se deberá implementar un método denominado "**prueba**" que permite verificar que los datos hayan sido colocados en la cola correctamente, para lo cual el método imprimirá todos los datos de la cola, este método no podrá ser utilizado para la atención. La plantilla solo podrá manipular los datos a través de operadores sobrecargados, NO de otro tipo de método. ***Solo el método para la llegada y la prueba de la cola podrán recorrer parte o toda la cola***. **La cola no podrá ser manipulada como una lista simplemente ligada**.

La plantilla deberá implementarse obligatoriamente mediante el siguiente diseño:



Para probar la plantilla deberá definir una **clase Dato** que contenga dos atributos enteros, en primero guardará la prioridad, y el segundo el dato propiamente dicho. También definirá una clase Prueba que tenga como atributo un objeto de clase **PriorityQueue**. **Deberá respetar los nombres dados a las clases y atributos, esto se tomará en cuenta en la calificación**.

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.

b) (7 puntos) Elabore un proyecto denominado **"Pacientes_Parte02B"**. Copie en él las plantillas 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**.

El proyecto deberá solucionar el problema siguiente, se tiene un archivo de tipo CSV con la siguiente información:

Parte02_ColaDeAtencion.csv

```
48308899,Zkenia Chung Higa,24,PQ5152C,E,Aldo Colado,12484697,Angel Kawa Lu,70,OH6720V,A,37.94,…
34417318,Delia Estefania Cabrera Moreno,28,AQ5291Y,U,15.18,17.49, …
…
```

El archivo está compuesto por varias líneas, contienen los datos de los pacientes que llegan a una institución de salud para ser atendidos. En cada línea puede haber más de un paciente. Los pacientes pueden ser de tres tipos diferentes: de **E**mergencia, de **U**rgencia o **A**mbulatorio.

Los datos de cada paciente estarán dados por 5 datos fijos: DNI, Nombre y edad del paciente, seguido del código del médico que lo atenderá y del tipo de paciente (caracter: 'E', 'U' o 'A'). Además, un paciente ambulatorio tiene un dato adicional, un valor de punto flotante que es el porcentaje de la tarifa de atención que pagará por tener seguro. Un paciente de urgencia tendrá dos datos adicionales, el porcentaje que pagará por seguro y un porcentaje adicional de descuento por su situación de urgencia. Los pacientes de emergencia deben proporcionar el nombre y número de teléfono de una persona de referencia a la que se pueda acudir en caso se requiera.

Para manejar los pacientes, el proyecto deberá definir una clase **PacientePQ** la cual estará definida de la siguiente manera:

Clase PacientePQ	
int dni;	double porcSeguro;
char *nombre;	double porcUrgencia;
int edad,	char *nombreRef;
char*codMed;	int telefRef;
char tipo;	

Debe respetar los nombres de la clase y atributos, no puede agregar más atributos. En esta clase deberá definir el constructor por defecto, constructor copia, destructor y todos los métodos que se requieran. Solo se llenarán los atributos que sean necesarios para el paciente.
Adicionalmente se cuenta con un archivo de textos como se muestra a continuación;

Parte02-TarifaPorConsulta.txt

```
FQ2435 Pablo/Quispe 356.79
NG1111 Naomi/Guzman 271.00
… …
```

En cada línea aparece el código y nombre del médico y la tarifa por su consulta.

El proyecto deberá definir una clase denominada **AtencionDeCitas**, la cual estará definida de la siguiente manera (no podrá modificar ni agregar más atributos):

```
Clase AtencionDeCitas
 class PriorityQueue <class PacientePQ> pacientes;
 list<class Tarifa> tarifas;
```

La clase **Tarifa** tendrá como atributos una cadena (char*) con el código del médico y un valor de punto flotante con la tarifa. En esta clase deberá definir todos los métodos, constructores y destructores que se requieran.

La clase **AtencionDeCitas** deberá contar con los siguientes métodos:

leerPacientes: recibirá como parámetro una cadena de caracteres con el nombre del archivo con los pacientes, el método deberá llenar la **PQ** con todos los pacientes. Los pacientes de emergencia tendrán mayor prioridad que los de urgencia y estos mayor que los ambulatorios, además dentro de una misma prioridad los adultos mayores (edad mayor de 65años) tendrán mayor prioridad que los niños (edad menor de 18 años) y estos, mayor que el resto, se sugiere que se defina un método que dé un valor a la prioridad del paciente.

leerTarifas: recibirá como parámetro una cadena de caracteres con el nombre del archivo con las tarifas, deberá cargar el contenedor con los códigos y tarifas. No podrá colocar o mover estos datos a otro tipo de contenedor, arreglo o estructura.

reporteDeAtenciones: recibirá como parámetros una cadena de caracteres con el nombre del archivo de reporte, este método, empleando el método atención (desencolar), deberá imprimir un reporte como el que se indica a continuación:

```
                                ATENCIONES MEDICAS
================================================================================
Codigo     Paciente                                 Edad    Medico
--------------------------------------------------------------------------------
35461278   Juan Perez Gomez                          77     FQ2345
           Referencia: Carlos Perez            Telefono: 991234421
           Tarifa del Medico: 356.79               Pago:      0.00
--------------------------------------------------------------------------------
…
--------------------------------------------------------------------------------
76412542   Pablo Sanchez Gonzales                    12     HG1289
           % por seguro: 23.05          % por urgencia:     10.00
           Tarifa del Medico: 399.99               Pago:     82.98
--------------------------------------------------------------------------------
…
--------------------------------------------------------------------------------
91122538   Ana Perez Garcia                          23     MG2251
           % por seguro: 37.94
           Tarifa del Medico: 569.85               Pago:    216.20
--------------------------------------------------------------------------------
…
```

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

Al finalizar la práctica, 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 esta este laboratorio.

Profesor del curso: Rony Cueva
Miguel Guanira

San Miguel, 28 de diciembre del 2020.