PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

LENGUAJES DE PROGRAMACIÓN 1
6ta práctica (tipo b)
Segundo Semestre 2020

Indicaciones Generales:

- Duración: 110 minutos.

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 objetos** (con excepción de los elementos de iostream, iomanip y fstream). 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. **Tampoco podrá hacer uso de plantillas.**
- 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 **main** de cada proyecto y el **código contenido en él solo podrá estar conformado por tareas implementadas como funciones**. 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 CALIFICARÁN AQUELLAS FUNCIONES DESARROLLADAS EN EL MISMO ARCHIVO QUE LA FUNCIÓN main.
- El código comentado NO SE CALIFICARÁ.
- Los proyectos deben obligatoriamente desarrollarse en NetBeans bajo el sistema operativo Windows. No se revisarán los proyectos desarrollados en otros sistemas operativos o IDEs.
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40% de puntaje de la pregunta. Los que no muestres resultados o que estos no sean coherentes en base al 60%.
- **Cada proyecto debe ser elaborado en un proyecto independiente, de no acatar esta restricción solo se calificará una pregunta.**

  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

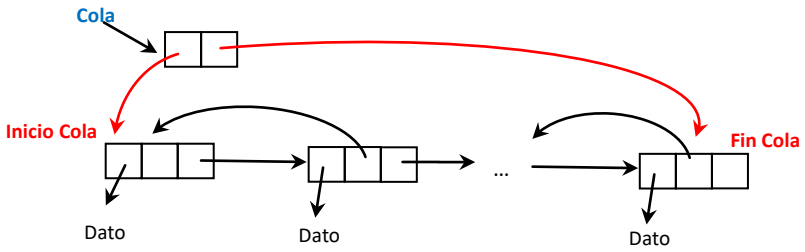
- **Puntaje total**: 20 puntos.

Problema

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).

Se desea elaborar una biblioteca de funciones denominada "**PriorityQueu**" (archivos .cpp y .h) que permita crear y manejar una cola de prioridad (**PQ**) genérica de datos. La biblioteca deberá manejar

obligatoriamente punteros a funciones y permitirá crear una **PQ** implementando operaciones de llegada (encolar), atención (desencolar) de datos[1] y verificación si la cola está o no vacía. También se deberá implementar una función denominada "**prueba**" que permite verificar que los datos hayan sido colocados en la cola correctamente, para lo cual la función imprimirá todos los datos de la cola, esta función no podrá ser utilizada para la atención. ***Solo la función para llegada y la que prueba la cola podrán recorrer parte o toda la cola***. **La cola no podrá ser manipulada como una lista simplemente ligada**. La misma debe tener obligatoriamente la siguiente estructura:



Como se puede observar los datos que llegan deben colocarse a partir del punto denominado "**Fin Cola**" y la atención se realizará a partir de del punto denominado "**Inicio Cola**" y la forma de llegar a esos puntos solo se podrá realizar a través del puntero "**Cola**". Además cada nodo será implementado con **un doble enlace**. La biblioteca solo debe contener las funciones que manejen de manera genérica la cola, las especificaciones propias de los datos que se estén trabajando deben hacerse en una biblioteca independiente.

Pregunta 1 (10 puntos)

Elabore un proyecto denominado **"PriorityQueuePruebaLab06"** para desarrollar y probar la biblioteca genérica. Para esta tarea primero deberá leer un archivo de texto que contiene solo pares de números enteros, cada par indicará un dato (como el ejemplo del primer párrafo del problema) y proceder a colocarlos en la cola según su prioridad, luego deberá verificar que los datos han sido colocados de manera correcta para lo cual debe usar la función "**prueba**", finalmente deberá volver a imprimir los datos de la cola pero está vez haciendo uso de la función de atención.

Pregunta 2 (10 puntos)

Una vez comprobado el buen funcionamiento de su biblioteca, elabore un proyecto denominado **"PriorityQueuePacientesLab06"** y en él copie la biblioteca "**PriorityQueu**" (archivos .cpp y .h) elaborada en la pregunta anterior. El proyecto deberá solucionar el problema siguiente:

Se tiene un archivo de tipo CSV con la siguiente información:

```
35461278,Juan Perez Gomez,E,3345,67548237,Maria Carpio Quispe,A,2251,25.65,…
76412542,Pablo Sanchez Gonzales,U,1289,23.05,10.0, …
…
```

El archivo, 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 estará dado por 4 datos fijos: DNI, Nombre, tipo de paciente (caracter: 'E', 'U' o 'A') y el código de la espacialidad que quiere atenderse. 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 no tienen datos adicionales puesto que no pagan.

El proyecto deberá leer todos los pacientes del archivo y colocarlos en la PQ, la primera prioridad la tendrán los pacientes de emergencia, luego los de urgencias y finalmente los ambulatorios. Luego, el

[1] La atención implica eliminar el nodo del inicio de la cola y entregar un puntero al dato.

programa atenderá uno a uno los datos de cada paciente mostrándolos en un reporte en el orden en que fueron atendidos con el pago realizado, como se muestra a continuación, los espacios de memoria asignados deben ser exactos.

```
                              ATENCIÓN DE PACIENTES
  ================================================================================
 No.    DNI        Paciente             Especialidad     Pago       Tipo
  1)    34523121   Pedro Aria Roca         5674            0.00     Emergencia
  2)    36458231   Rosa Quispe Garma       3354            0.00     Emergencia
  …     …          …                         …             …        …
 35)    24712932   Patricia Ludena Ho      3354          120.70     Urgencia
 36)    14253723   Jose Daso Lopez         1234           39.45     Urgencia
  …     …          …                         …             …        …
104)    7345129    Felipe Castro Vera      3354           78.34     Ambulatorio
105)    4236124    Sara Ruiz Lara          1923          280.00     Ambulatorio
================================================================================
Total cobrado: …
```

Para la determinación del pago se contará con un archivo de textos similar al siguiente:

```
1234   275.89
3354   390.75
1923  1256.89
…
```

En cada línea se tendrá el código de la espacialidad seguido de la tarifa por consulta.

Los archivos solo se pueden leer una vez.

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 este laboratorio.

Profesores del curso: Miguel Guanira
Rony Cueva

San Miguel, 13 de noviembre del 2020.